Docket No.: 57454-335

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of	:	
Masato HAGIWARA, et al.	:	
Serial No.:	:	Group Art Unit:
Filed: February 13, 2002	:	Examiner:

For: PROGRAM EXECUTION DEVICE OPERATING BASED ON COMPRESSED CODE

CLAIM OF PRIORITY AND
TRANSMITTAL OF CERTIFIED PRIORITY DOCUMENT

Commissioner for Patents
Washington, DC 20231

Sir:

In accordance with the provisions of 35 U.S.C. 119, Applicants hereby claim the priority of:

Japanese Patent Application No. 2001-124824(P), filed April 23, 2001

cited in the Declaration of the present application. A certified copy is submitted herewith.

Respectfully submitted,

MCDERMOTT, WILL & EMERY

Stephen A. Becker
Registration No. 26,527

600 13th Street, N.W.
Washington, DC 20005-3096
(202) 756-8000 SAB:mlw
Date: February 13, 2002
Facsimile: (202) 756-8087

57454-335
Masato HAGIWARA et al.
February 13, 2002

日　本　国　特　許　庁
JAPAN PATENT OFFICE

McDermott, Will & Emery

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２００１年　４月２３日

出願番号
Application Number: 特願２００１－１２４８２４

出願人
Applicant(s): 三菱電機株式会社

２００１年　５月１８日

特許庁長官
Commissioner,
Japan Patent Office 及川耕造

出証番号　出証特２００１－３０４２６６８

【書類名】　特許願

【整理番号】　530049JP01

【提出日】　平成13年 4月23日

【あて先】　特許庁長官殿

【国際特許分類】　G06F　9/45

【発明者】

【住所又は居所】　東京都千代田区丸の内二丁目２番３号　三菱電機株式会社内

【氏名】　萩原　正人

【発明者】

【住所又は居所】　東京都千代田区丸の内二丁目２番３号　三菱電機株式会社内

【氏名】　吉田　豊彦

【発明者】

【住所又は居所】　東京都千代田区丸の内二丁目２番３号　三菱電機株式会社内

【氏名】　坂本　守

【特許出願人】

【識別番号】　000006013

【氏名又は名称】　三菱電機株式会社

【代理人】

【識別番号】　100064746

【弁理士】

【氏名又は名称】　深見　久郎

【選任した代理人】

【識別番号】　100085132

【弁理士】

【氏名又は名称】　森田　俊雄

【選任した代理人】

【識別番号】　100091409

【弁理士】

【氏名又は名称】　伊藤　英彦

【選任した代理人】

【識別番号】　100096781

【弁理士】

【氏名又は名称】　堀井　豊

【選任した代理人】

【識別番号】　100096792

【弁理士】

【氏名又は名称】　森下　八郎

【手数料の表示】

【予納台帳番号】　008693

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　プログラム実行装置および方法

【特許請求の範囲】

【請求項１】　所定の言語で記述されたプログラムを実行するプログラム実行装置であって、

前記プログラムの、所定の単位毎に圧縮されたコードを格納する圧縮コード格納部と、

前記圧縮コード格納部に接続され、前記圧縮コード格納部から前記圧縮されたコードを読出して伸張するための伸張手段と、

前記伸張手段に接続され、前記伸張手段により伸張されたコードを格納するコード格納部と、

前記コード格納部に接続され、伸張された前記コードを解釈し、実行するためのインタープリタ手段とを含む、プログラム実行装置。

【請求項２】　前記所定の言語はオブジェクト指向言語であり、前記所定の単位は、メソッドである、請求項１に記載のプログラム実行装置。

【請求項３】　前記所定の単位は、プログラム中で分岐を含まない一連の命令である、請求項１に記載のプログラム実行装置。

【請求項４】　前記所定の単位は、命令である、請求項１に記載のプログラム実行装置。

【請求項５】　さらに、前記圧縮コード格納部に接続され、圧縮された前記コードに基づいて、前記コードの圧縮方式を判定するための圧縮方式判定手段を含み、

前記伸張手段は、前記圧縮コード格納部、前記コード格納部および前記圧縮方式判定手段に接続され、前記圧縮方式判定手段の出力に基づいて、圧縮された前記コードを伸張し、前記コード格納部に格納するための手段を含む、請求項１に記載のプログラム実行装置。

【請求項６】　オブジェクト指向言語で記述されたプログラムを実行するプログラム実行装置であって、前記プログラムは前記プログラム実行装置にとってネイティブなコード以外のコードで記述されており、

メソッド毎に圧縮された前記プログラムのコードを格納する圧縮コード格納部と、

前記圧縮コード格納部に接続され、圧縮された前記コードを伸張するための伸張手段と、

前記伸張手段に接続され、伸張された前記コードをネイティブコードに変換するための変換手段と、

前記変換手段に接続され、前記変換手段により出力されるネイティブコードを格納するネイティブコード格納部と、

前記ネイティブコード格納部に接続され、前記ネイティブコードを実行するためのネイティブコード実行手段とを含む、プログラム実行装置。

【請求項７】　前記ネイティブコード格納部は、キャッシュメモリよりなる、請求項６に記載のプログラム実行装置。

【請求項８】　オブジェクト指向言語で記述されたプログラムを実行するプログラム実行装置であって、

前記プログラムのメソッドのコードを格納するコード格納部と、

メソッドのネイティブコードを格納するネイティブコード格納部と、

メソッドのネイティブコードを圧縮した圧縮ネイティブコードを格納する圧縮ネイティブコード格納部と、

前記ネイティブコード格納部に接続され、前記ネイティブコード格納部に所望のメソッドのネイティブコードが格納されているか否かを判断するための第１の判断手段と、

前記圧縮ネイティブコード格納部に接続され、前記圧縮ネイティブコード格納部に前記所望のメソッドの圧縮ネイティブコードが格納されているか否かを判断する第２の判断手段と、

前記第１および第２の判断手段、前記圧縮ネイティブコード格納部、前記コード格納部および前記ネイティブコード格納部に接続され、前記第１および第２の判断手段の出力に基づいて、前記圧縮ネイティブコード格納部に格納された圧縮ネイティブコードの伸張または前記コード格納部に格納されたコードのネイティブコードへの変換を選択的に実行し、得られたネイティブコードを前記ネイティ

ブコード格納部へ格納するための、ネイティブコード格納制御手段と、

前記ネイティブコード格納部に接続され、前記ネイティブコード格納部に格納されたネイティブコードを実行するためのネイティブコード実行手段と、

前記第２の判断手段、前記ネイティブコード格納部および前記圧縮ネイティブコード格納部に接続され、前記第２の判断手段の出力に基づいて、実行されたネイティブコードを圧縮し、前記圧縮ネイティブコード格納部へ格納するためのネイティブコード圧縮格納手段とを含む、プログラム実行装置。

【請求項９】　さらに、前記圧縮ネイティブコード格納部に格納されている圧縮ネイティブコードの圧縮方式をメソッド毎に格納する圧縮方式格納部を含み、

前記ネイティブコード格納制御手段は、前記第１および第２の判断手段、前記圧縮ネイティブコード格納部、前記コード格納部、前記ネイティブコード格納部および前記圧縮方式格納部に接続され、前記第１および第２の判断手段の出力に基づいて、前記圧縮方式格納部に格納された圧縮方式に従った前記圧縮ネイティブコード格納部に格納された圧縮ネイティブコードの伸張または前記コード格納部に格納されたコードのネイティブコードへの変換を選択的に実行し、得られたネイティブコードを前記ネイティブコード格納部へ格納するための手段を含み、

前記ネイティブコード圧縮格納手段は、前記第２の判断手段、前記ネイティブコード格納部、前記圧縮ネイティブコード格納部および前記圧縮方式格納部に接続され、前記第２の判断手段の出力に基づいて、実行されたネイティブコードを予め定められた方法に従い定められる圧縮方式により圧縮し、圧縮ネイティブコードを前記圧縮ネイティブコード格納部へ格納し、前記圧縮方式を前記圧縮方式格納部へ格納するための手段を含む、請求項８に記載のプログラム実行装置。

【請求項１０】　前記ネイティブコード圧縮格納手段は、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、最も古くネイティブコードに変換されたものから優先的に圧縮する、請求項８に記載のプログラム実行装置。

【請求項１１】　前記ネイティブコード圧縮格納手段は、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、実行頻度の低

いものから優先的に圧縮する、請求項８に記載のプログラム実行装置。

【請求項１２】　前記ネイティブコード圧縮格納手段は、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、サイズの大きいものから優先的に圧縮する、請求項８に記載のプログラム実行装置。

【請求項１３】　前記ネイティブコード圧縮格納手段は、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、最も圧縮率が大きいものから優先的に圧縮する、請求項８に記載のプログラム実行装置。

【請求項１４】　前記ネイティブコード圧縮格納手段は、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最も実行頻度が低い圧縮ネイティブコードを削除する、請求項８に記載のプログラム実行装置。

【請求項１５】　前記ネイティブコード圧縮格納手段は、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最もサイズが大きい圧縮ネイティブコードを削除する、請求項８に記載のプログラム実行装置。

【請求項１６】　前記ネイティブコード圧縮格納手段は、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最も圧縮率が低い圧縮ネイティブコードを削除する、請求項８に記載のプログラム実行装置。

【請求項１７】　前記ネイティブコード圧縮格納手段は、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最も古くに圧縮された圧縮ネイティブコードを削除する、請求項８に記載のプログラム実行装置。

【請求項１８】　オブジェクト指向言語で記述されたプログラムを実行するプログラム実行方法であって、

所定の単位毎に圧縮されたコードを伸張するステップと、

伸張されたコードを解釈し、実行するステップとを含む、プログラム実行方法。

【請求項１９】　前記所定の単位は、メソッドである、請求項１８に記載の

プログラム実行方法。

【請求項２０】　前記所定の単位は、プログラム中で分岐を含まない一連の命令である、請求項１８に記載のプログラム実行方法。

【請求項２１】　前記所定の単位は、命令である、請求項１８に記載のプログラム実行方法。

【請求項２２】　さらに、所定の単位毎に圧縮されたコードの圧縮方式を判定するステップを含み、

前記伸張するステップは、判定された圧縮方式に基づいて、所定の単位毎に圧縮されたコードを伸張するステップを含む、請求項１８に記載のプログラム実行方法。

【請求項２３】　オブジェクト指向言語で記述されたプログラムを実行するプログラム実行方法であって、

メソッド毎に圧縮されたコードを伸張するステップと、

伸張された前記コードをネイティブコードに変換するステップと、

前記ネイティブコードを実行するステップとを含む、プログラム実行方法。

【請求項２４】　オブジェクト指向言語で記述されたプログラムを実行するプログラム実行装置で用いられるプログラム実行方法であって、

前記プログラム実行装置は、前記プログラムのメソッドのコードを格納するコード格納部と、メソッドのネイティブコードを格納するネイティブコード格納部と、メソッドのネイティブコードを圧縮した圧縮ネイティブコードを格納する圧縮ネイティブコード格納部とを含み、

前記ネイティブコード格納部に所望のメソッドのネイティブコードが格納されているか否かを判断するステップと、

前記圧縮ネイティブコード格納部に前記所望のメソッドの圧縮ネイティブコードが格納されているか否かを判断するステップと、

前記ネイティブコードの格納結果および前記圧縮ネイティブコードの格納結果に基づいて、前記圧縮ネイティブコード格納部に格納された圧縮ネイティブコードの伸張または前記コード格納部に格納されたコードのネイティブコードへの変換を選択的に実行し、得られたネイティブコードを前記ネイティブコード格納部

へ格納するステップと、

前記ネイティブコード格納部に格納されたネイティブコードを実行するステップと、

前記圧縮ネイティブコードの格納結果に基づいて、実行されたネイティブコードを圧縮し、前記圧縮ネイティブコード格納部へ格納するステップとを含む、プログラム実行方法。

【請求項２５】　前記プログラム実行装置は、さらに、前記圧縮ネイティブコード格納部に格納されている圧縮ネイティブコードの圧縮方式をメソッド毎に格納する圧縮方式格納部を含み、

前記ネイティブコードを前記ネイティブコード格納部へ格納する前記ステップは、前記ネイティブコードの格納結果および前記圧縮ネイティブコードの格納結果に基づいて、前記圧縮方式格納部に格納された圧縮方式に従った前記圧縮ネイティブコード格納部に格納された圧縮ネイティブコードの伸張または前記コード格納部に格納されたコードのネイティブコードへの変換を選択的に実行し、得られたネイティブコードを前記ネイティブコード格納部へ格納するステップを含み、

前記圧縮ネイティブコード格納部へ格納する前記ステップは、前記圧縮ネイティブコードの格納結果に基づいて、実行されたネイティブコードを予め定められた方法に従い定められる圧縮方式により圧縮し、圧縮ネイティブコードを前記圧縮ネイティブコード格納部へ格納し、前記圧縮方式を前記圧縮方式格納部へ格納するステップを含む、請求項２４に記載のプログラム実行方法。

【請求項２６】　前記圧縮ネイティブコード格納部へ格納する前記ステップは、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、最も古くネイティブコードに変換されたものから優先的に圧縮する、請求項２４に記載のプログラム実行方法。

【請求項２７】　前記圧縮ネイティブコード格納部へ格納する前記ステップは、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、実行頻度の低いものから優先的に圧縮する、請求項２４に記載のプログラム実行方法。

【請求項２８】　前記圧縮ネイティブコード格納部へ格納する前記ステップは、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、サイズの大きいものから優先的に圧縮する、請求項２４に記載のプログラム実行方法。

【請求項２９】　前記圧縮ネイティブコード格納部へ格納する前記ステップは、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、最も圧縮率が大きいものから優先的に圧縮する、請求項２４に記載のプログラム実行方法。

【請求項３０】　前記圧縮ネイティブコード格納部へ格納する前記ステップは、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最も実行頻度が低い圧縮ネイティブコードを削除する、請求項２４に記載のプログラム実行方法。

【請求項３１】　前記圧縮ネイティブコード格納部へ格納する前記ステップは、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最もサイズが大きい圧縮ネイティブコードを削除する、請求項２４に記載のプログラム実行方法。

【請求項３２】　前記圧縮ネイティブコード格納部へ格納する前記ステップは、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最も圧縮率が低い圧縮ネイティブコードを削除する、請求項２４に記載のプログラム実行方法。

【請求項３３】　前記圧縮ネイティブコード格納部へ格納する前記ステップは、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最も古くに圧縮された圧縮ネイティブコードを削除する、請求項２４に記載のプログラム実行方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、プログラム実行装置および方法に関し、特に、データを格納するメモリの容量が小さいプログラム実行装置および方法に関する。

【０００２】

【従来の技術】

Ｊａｖａ（Ｒ）（米国サンマイクロシステムズ社の登録商標）言語で記述されたプログラムは、実行に先だってバイトコードと呼ばれるプラットフォームに依存しないコードにコンパイルされ、配布される。Ｊａｖａ（Ｒ）仮想マシンは、インタプリタを用いてバイトコードを１命令ずつ解釈しながら実行することにより、プログラムの実行を行なう。最近では、バイトコードをそのまま実行するのではなく、ＪＩＴ（Just-in-time Compiler）を用いて、バイトコードをネイティブコードに変換し、プログラムを高速に実行できるようにしたものもある。

【０００３】

このような、Ｊａｖａ（Ｒ）言語で記述されたプログラムは、携帯電話機、ＰＤＡ（Personal Digital Assistants）または情報家電などの組込み機器においても広く用いられている。

【０００４】

【発明が解決しようとする課題】

しかし、組込み機器では、使用できるＲＯＭ（Read Only Memory）またはＲＡＭ（Random Access Memory）の容量が通常のコンピュータに比べて低い。このため、ステップ数の大きなプログラムを実行できないという問題がある。

【０００５】

本発明は上述の課題を解決するためになされたもので、その目的は、必要なメモリの容量が小さいプログラム実行装置および方法を提供することである。

【０００６】

【課題を解決するための手段】

本発明のある局面に従うプログラム実行装置は、所定の言語で記述されたプログラムを実行する。プログラム実行装置は、前記プログラムの、所定の単位毎に

圧縮されたコードを格納する圧縮コード格納部と、前記圧縮コード格納部に接続され、前記圧縮コード格納部から前記圧縮されたコードを読出して伸張するための伸張手段と、前記伸張手段に接続され、前記伸張手段により伸張されたコードを格納するコード格納部と、前記コード格納部に接続され、伸張された前記コードを解釈し、実行するためのインタープリタ手段とを含む。

【０００７】

所定の単位毎にコードが圧縮されて圧縮コード格納部に格納されている。このため、圧縮コード格納部の容量を小さくすることができ、プログラム実行装置で必要とするメモリの容量を小さくすることができる。

【０００８】

好ましくは、前記所定の言語はオブジェクト指向言語であり、前記所定の単位は、メソッドである。

【０００９】

さらに好ましくは、前記所定の単位は、プログラム中で分岐を含まない一連の命令である。

【００１０】

コード格納部には、分岐を含まない一連の命令のコードが格納される。このため、メソッドのコードを格納する場合に比べ、コード格納部の容量を小さくすることができ、プログラム実行装置で必要とするメモリの容量を小さくすることができる。

【００１１】

さらに好ましくは、前記所定の単位は、命令である。

コード格納部には、命令のコードが格納される。このため、メソッドのコードを格納する場合に比べ、コード格納部の容量を小さくすることができ、プログラム実行装置で必要とするメモリの容量を小さくすることができる。

【００１２】

さらに好ましくは、プログラム実行装置は、さらに、前記圧縮コード格納部に接続され、圧縮された前記コードに基づいて、前記コードの圧縮方式を判定するための圧縮方式判定手段を含む。前記伸張手段は、前記圧縮コード格納部、前記

コード格納部および前記圧縮方式判定手段に接続され、前記圧縮方式判定手段の出力に基づいて、圧縮された前記コードを伸張し、前記コード格納部に格納するための手段を含む。

【００１３】

メソッド毎にコードの圧縮方式をユーザが選択することができ、最適な圧縮方式を採用することができる。このため、圧縮コード格納部の容量を小さくすることができ、プログラム実行装置で必要とするメモリの容量を小さくすることができる。

【００１４】

本発明の他の局面に従うプログラム実行装置は、オブジェクト指向言語で記述されたプログラムを実行する。前記プログラムは前記プログラム実行装置にとってネイティブなコード以外のコードで記述されている。プログラム実行装置は、メソッド毎に圧縮された前記プログラムのコードを格納する圧縮コード格納部と、前記圧縮コード格納部に接続され、圧縮された前記コードを伸張するための伸張手段と、前記伸張手段に接続され、伸張された前記コードをネイティブコードに変換するための変換手段と、前記変換手段に接続され、前記変換手段により出力されるネイティブコードを格納するネイティブコード格納部と、前記ネイティブコード格納部に接続され、前記ネイティブコードを実行するためのネイティブコード実行手段とを含む。

【００１５】

メソッドのコードが圧縮されて圧縮コード格納部に格納されている。このため、圧縮コード格納部の容量を小さくすることができ、プログラム実行装置で必要とするメモリの容量を小さくすることができる。

【００１６】

好ましくは、前記ネイティブコード格納部は、キャッシュメモリよりなる。

ネイティブコードをキャッシュメモリ内に格納することにより、プログラムの実行速度を向上させることができる。

【００１７】

本発明のさらに他の局面に従うプログラム実行装置は、オブジェクト指向言語

で記述されたプログラムを実行する。プログラム実行装置は、前記プログラムのメソッドのコードを格納するコード格納部と、メソッドのネイティブコードを格納するネイティブコード格納部と、メソッドのネイティブコードを圧縮した圧縮ネイティブコードを格納する圧縮ネイティブコード格納部と、前記ネイティブコード格納部に接続され、前記ネイティブコード格納部に所望のメソッドのネイティブコードが格納されているか否かを判断するための第１の判断手段と、前記圧縮ネイティブコード格納部に接続され、前記圧縮ネイティブコード格納部に前記所望のメソッドの圧縮ネイティブコードが格納されているか否かを判断する第２の判断手段と、前記第１および第２の判断手段、前記圧縮ネイティブコード格納部、前記コード格納部および前記ネイティブコード格納部に接続され、前記第１および第２の判断手段の出力に基づいて、前記圧縮ネイティブコード格納部に格納された圧縮ネイティブコードの伸張または前記コード格納部に格納されたコードのネイティブコードへの変換を選択的に実行し、得られたネイティブコードを前記ネイティブコード格納部へ格納するための、ネイティブコード格納制御手段と、前記ネイティブコード格納部に接続され、前記ネイティブコード格納部に格納されたネイティブコードを実行するためのネイティブコード実行手段と、前記第２の判断手段、前記ネイティブコード格納部および前記圧縮ネイティブコード格納部に接続され、前記第２の判断手段の出力に基づいて、実行されたネイティブコードを圧縮し、前記圧縮ネイティブコード格納部へ格納するためのネイティブコード圧縮格納手段とを含む。

【００１８】

コードがネイティブコードに変換された後、圧縮されて圧縮ネイティブコード格納部に格納される。このため、ＪＩＴを用いた従来の装置に比べ、ネイティブコード格納部の容量を小さくすることができる。

【００１９】

好ましくは、プログラム実行装置は、さらに、前記圧縮ネイティブコード格納部に格納されている圧縮ネイティブコードの圧縮方式をメソッド毎に格納する圧縮方式格納部を含む。前記ネイティブコード格納制御手段は、前記第１および第２の判断手段、前記圧縮ネイティブコード格納部、前記コード格納部、前記ネイ

ティブコード格納部および前記圧縮方式格納部に接続され、前記第１および第２の判断手段の出力に基づいて、前記圧縮方式格納部に格納された圧縮方式に従った前記圧縮ネイティブコード格納部に格納された圧縮ネイティブコードの伸張または前記コード格納部に格納されたコードのネイティブコードへの変換を選択的に実行し、得られたネイティブコードを前記ネイティブコード格納部へ格納するための手段を含む。前記ネイティブコード圧縮格納手段は、前記第２の判断手段、前記ネイティブコード格納部、前記圧縮ネイティブコード格納部および前記圧縮方式格納部に接続され、前記第２の判断手段の出力に基づいて、実行されたネイティブコードを予め定められた方法に従い定められる圧縮方式により圧縮し、圧縮ネイティブコードを前記圧縮ネイティブコード格納部へ格納し、前記圧縮方式を前記圧縮方式格納部へ格納するための手段を含む。

【００２０】

メソッド毎に最適な圧縮方式を用いてネイティブコードの圧縮が行なわれる。このため、圧縮ネイティブコード格納部の容量を小さくすることができる。

【００２１】

さらに好ましくは、前記ネイティブコード圧縮格納手段は、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、最も古くネイティブコードに変換されたものから優先的に圧縮する。

【００２２】

さらに好ましくは、前記ネイティブコード圧縮格納手段は、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、実行頻度の低いものから優先的に圧縮する。

【００２３】

さらに好ましくは、前記ネイティブコード圧縮格納手段は、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、サイズの大きいものから優先的に圧縮する。

【００２４】

さらに好ましくは、前記ネイティブコード圧縮格納手段は、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、最も圧縮率が

大きいものから優先的に圧縮する。

【００２５】

さらに好ましくは、前記ネイティブコード圧縮格納手段は、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最も実行頻度が低い圧縮ネイティブコードを削除する。

【００２６】

さらに好ましくは、前記ネイティブコード圧縮格納手段は、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最もサイズが大きい圧縮ネイティブコードを削除する。

【００２７】

さらに好ましくは、前記ネイティブコード圧縮格納手段は、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最も圧縮率が低い圧縮ネイティブコードを削除する。

【００２８】

さらに好ましくは、前記ネイティブコード圧縮格納手段は、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最も古くに圧縮された圧縮ネイティブコードを削除する。

【００２９】

本発明のさらに他の局面に従うプログラム実行方法は、オブジェクト指向言語で記述されたプログラムを実行する。プログラム実行方法は、所定の単位毎に圧縮されたコードを伸張するステップと、伸張されたコードを解釈し、実行するステップとを含む。

【００３０】

圧縮されたコードを伸張してから、コードの解釈および実行が行われる。このようにコードは予め圧縮されているため、小さなメモリ容量の装置においても、

プログラムを実行することができる。

【００３１】

好ましくは、前記所定の単位は、メソッドである。

さらに好ましくは、前記所定の単位は、プログラム中で分岐を含まない一連の命令である。

【００３２】

分岐を含まない一連の命令単位でコードが圧縮される。このため、メソッド単位でコードが圧縮される場合に比べ、伸張されたコードを記憶するメモリの容量が少なくてすむ。

【００３３】

さらに好ましくは、前記所定の単位は、命令である。

命令単位でコードが圧縮される。このため、メソッド単位でコードが圧縮される場合に比べ、伸張されたコードを記憶するメモリの容量が少なくてすむ。

【００３４】

さらに好ましくは、プログラム実行方法は、さらに、所定の単位毎に圧縮されたコードの圧縮方式を判定するステップを含む。前記伸張するステップは、判定された圧縮方式に基づいて、所定の単位毎に圧縮されたコードを伸張するステップを含む。

【００３５】

メソッド毎にコードの圧縮方式をユーザが選択することができ、最適な圧縮方式を採用することができる。このため、圧縮されたコードを格納するメモリの容量を小さくすることができる。

【００３６】

本発明のさらに他の局面に従うプログラム実行方法は、オブジェクト指向言語で記述されたプログラムを実行する。プログラム実行方法は、メソッド毎に圧縮されたコードを伸張するステップと、伸張された前記コードをネイティブコードに変換するステップと、前記ネイティブコードを実行するステップとを含む。

【００３７】

メソッドのコードが圧縮されている。このため、メソッドのコードを格納する

メモリの容量を小さくすることができる。

【００３８】

本発明のさらに他の局面に従うプログラム実行方法は、オブジェクト指向言語で記述されたプログラムを実行するプログラム実行装置で用いられる。前記プログラム実行装置は、前記プログラムのメソッドのコードを格納するコード格納部と、メソッドのネイティブコードを格納するネイティブコード格納部と、メソッドのネイティブコードを圧縮した圧縮ネイティブコードを格納する圧縮ネイティブコード格納部とを含む。プログラム実行方法は、前記ネイティブコード格納部に所望のメソッドのネイティブコードが格納されているか否かを判断するステップと、前記圧縮ネイティブコード格納部に前記所望のメソッドの圧縮ネイティブコードが格納されているか否かを判断するステップと、前記ネイティブコードの格納結果および前記圧縮ネイティブコードの格納結果に基づいて、前記圧縮ネイティブコード格納部に格納された圧縮ネイティブコードの伸張または前記コード格納部に格納されたコードのネイティブコードへの変換を選択的に実行し、得られたネイティブコードを前記ネイティブコード格納部へ格納するステップと、前記ネイティブコード格納部に格納されたネイティブコードを実行するステップと、前記圧縮ネイティブコードの格納結果に基づいて、実行されたネイティブコードを圧縮し、前記圧縮ネイティブコード格納部へ格納するステップとを含む。

【００３９】

コードがネイティブコードに変換された後、圧縮されて圧縮ネイティブコード格納部に格納される。このため、ＪＩＴを用いた従来の装置に比べ、ネイティブコード格納部の容量を小さくすることができる。

【００４０】

好ましくは、前記プログラム実行装置は、さらに、前記圧縮ネイティブコード格納部に格納されている圧縮ネイティブコードの圧縮方式をメソッド毎に格納する圧縮方式格納部を含む。前記ネイティブコードを前記ネイティブコード格納部へ格納する前記ステップは、前記ネイティブコードの格納結果および前記圧縮ネイティブコードの格納結果に基づいて、前記圧縮方式格納部に格納された圧縮方式に従った前記圧縮ネイティブコード格納部に格納された圧縮ネイティブコード

の伸張または前記コード格納部に格納されたコードのネイティブコードへの変換を選択的に実行し、得られたネイティブコードを前記ネイティブコード格納部へ格納するステップを含む。前記圧縮ネイティブコード格納部へ格納する前記ステップは、前記圧縮ネイティブコードの格納結果に基づいて、実行されたネイティブコードを予め定められた方法に従い定められる圧縮方式により圧縮し、圧縮ネイティブコードを前記圧縮ネイティブコード格納部へ格納し、前記圧縮方式を前記圧縮方式格納部へ格納するステップを含む。

【００４１】

メソッド毎に最適な圧縮方式を用いてネイティブコードの圧縮が行なわれる。このため、圧縮ネイティブコード格納部の容量を小さくすることができる。

【００４２】

さらに好ましくは、前記圧縮ネイティブコード格納部へ格納する前記ステップは、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、最も古くネイティブコードに変換されたものから優先的に圧縮する。

【００４３】

さらに好ましくは、前記圧縮ネイティブコード格納部へ格納する前記ステップは、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、実行頻度の低いものから優先的に圧縮する。

【００４４】

さらに好ましくは、前記圧縮ネイティブコード格納部へ格納する前記ステップは、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、サイズの大きいものから優先的に圧縮する。

【００４５】

さらに好ましくは、前記圧縮ネイティブコード格納部へ格納する前記ステップは、前記ネイティブコード格納部に格納されているメソッドのネイティブコードのうち、最も圧縮率が大きいものから優先的に圧縮する。

【００４６】

さらに好ましくは、前記圧縮ネイティブコード格納部へ格納する前記ステップは、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティ

ブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最も実行頻度が低い圧縮ネイティブコードを削除する。

【００４７】

さらに好ましくは、前記圧縮ネイティブコード格納部へ格納する前記ステップは、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最もサイズが大きい圧縮ネイティブコードを削除する。

【００４８】

さらに好ましくは、前記圧縮ネイティブコード格納部へ格納する前記ステップは、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最も圧縮率が低い圧縮ネイティブコードを削除する。

【００４９】

さらに好ましくは、前記圧縮ネイティブコード格納部へ格納する前記ステップは、圧縮ネイティブコードの格納領域に空きがない場合には、前記圧縮ネイティブコード格納領域に格納されているメソッドの圧縮ネイティブコードのうち、最も古くに圧縮された圧縮ネイティブコードを削除する。

【００５０】

【発明の実施の形態】

［実施の形態１］

本発明の第１の実施の形態に係るＪａｖａ（Ｒ）仮想マシンは、組込み機器を用いて実現される。

【００５１】

図１を参照して、組込み機器は、プログラムを解釈、実行するＣＰＵ（Central Processing Unit）２と、ＣＰＵ２で実行されるプログラムのバイトコードを記憶するＲＯＭ（Read Only Memory）４と、そのプログラムを実行するための各種データを記憶するＲＡＭ（Random Access Memory）８と、ユーザとの間でデータの入出力を行なうためのユーザインタフェース６と、ＣＰＵ２、ＲＯＭ４、ユーザインタフェース６およびＲＡＭ８を相互に接続するバスとを含む。

【００５２】

図２を参照して、ＲＯＭ４は、Ｊａｖａ（Ｒ）言語で記述されたプログラムのバイトコードを圧縮したもの（以下「圧縮バイトコード」と言う。）を格納する圧縮バイトコード格納領域１２と、その他のクラス情報を格納する領域１４とを含む。

【００５３】

図３を参照して、ＲＡＭ８は、オペレーティングシステムを格納するオペレーティングシステム格納領域１６と、バーチャルマシン（ＶＭ）モジュールを格納するＶＭモジュール格納領域１８と、データを圧縮するためのプログラムである圧縮モジュールを格納するための圧縮モジュール格納領域２４と、圧縮されたデータを伸張するためのプログラムである伸張モジュールを格納するための伸張モジュール格納領域２６と、バイトコードおよびネイティブコード等を格納するためのコード格納領域２８と、メソッドの状態を表わすメソッドステータスを格納するためのメソッドステータス格納領域３６とを含む。メソッドステータス格納領域３６はメソッド毎に設けられる。

【００５４】

ＶＭモジュール格納領域１８は、インタプリタを格納するためのインタプリタ格納領域２０と、ＪＩＴを格納するためのＪＩＴ格納領域２２とを含む。

【００５５】

コード格納領域２８は、伸張済みバイトコードを格納する伸張済みバイトコード格納領域３０と、ネイティブコードを格納するネイティブコード格納領域３２と、ネイティブコードを圧縮したもの（以下「圧縮ネイティブコード」と言う。）を格納する圧縮ネイティブコード格納領域３４とを含む。

【００５６】

メソッドステータス格納領域３６は、メソッドが実行された回数を格納する回数格納領域３８と、メソッドのサイズを格納するサイズ格納領域４０と、ネイティブコードが存在するか否かを示すネイティブコード有無フラグを格納するネイティブコード有無フラグ格納領域４２と、バイトコードがコンパイルされた時刻または順番を格納するコンパイル時刻（順番）格納領域４４と、圧縮に関する情

報を格納する圧縮情報格納領域４６とを含む。

【００５７】

圧縮情報格納領域４６は、メソッドが圧縮されているか否かを示す圧縮フラグを格納する圧縮フラグ格納領域４８と、圧縮されたメソッドが伸張されているか否かを示す伸張済みフラグを格納する伸張済みフラグ格納領域５０と、メソッドの圧縮方式を格納する圧縮方式格納領域５２と、圧縮の際の圧縮率を格納する圧縮率格納領域５４と、圧縮された時刻または順番を格納する圧縮時刻（順番）格納領域５６と、圧縮ネイティブコードのサイズを格納する圧縮サイズ格納領域５８とを含む。

【００５８】

図４を参照して、本実施の形態では、Ｊａｖａ（Ｒ）言語で記述されたプログラムは、メソッド１～４の４つのメソッドより構成されるものとして説明を行なう。プログラムがそれ以外の個数のメソッドより構成されている場合であっても、同様の動作を行なう。

【００５９】

すべてのメソッドは、メソッド毎に圧縮され、圧縮バイトコード格納領域１２に格納されている。

【００６０】

図５を参照して、上位のモジュールよりあるメソッドの呼出しがあった場合には、ＣＰＵ２は、呼出されたメソッドの伸張済みフラグ格納領域５０に格納されている伸張済みフラグを参照し、呼出されたメソッドのバイトコードが、伸張されており、伸張済みバイトコード格納領域３０に格納されているか否かを調べる（Ｓ２）。

【００６１】

バイトコードが伸張されていなければ（Ｓ２でＮＯ）、圧縮バイトコード格納領域１２に格納されたバイトコードが伸張され（Ｓ４）、伸張後のバイトコードが伸張済みバイトコード格納領域３０に格納される（Ｓ６）。次に、このメソッドの圧縮バイトコードが伸張済みであることを表わすため、伸張済みフラグの値がオンに変更される（Ｓ８）。

【００６２】

Ｓ８の処理の後、またはメソッドのバイトコードが伸張済みバイトコード格納領域３０に格納されている場合には（Ｓ２でＹＥＳ）、ＣＰＵ２はインタプリタを用いて、伸張済みのバイトコードを１命令ずつ解釈しながら実行する（Ｓ１０）。その後、呼出し側のモジュールへ復帰するための処理を行なう（Ｓ１２）。

【００６３】

なお、伸張済みバイトコード格納領域３０に書込まれていた伸張済みのバイトコードが削除または他のバイトコードの上書き等により失われる場合には、そのバイトコードに対応したメソッドの伸張済みフラグがオフに設定される。

【００６４】

上述の４つのメソッドがメソッド１、メソッド２、メソッド３、メソッド１、メソッド２、メソッド４、メソッド４、メソッド４の順に呼出されて、実行される場合を想定する。また、伸張済みバイトコード格納領域３０には、１つ分のメソッドのバイトコードしか格納できないとする。このとき、図６を参照して、ＣＰＵ２は以下の順序で処理を実行する。

【００６５】

メソッド１の圧縮バイトコードを伸張し、伸張済みフラグをオンにする（Ｓ２２）。メソッド１のバイトコードを１命令ずつ解釈しながら実行する（Ｓ２４）。メソッド２の圧縮バイトコードを伸張し、伸張済みフラグをオンにする（Ｓ２６）。メソッド２のバイトコードを１命令ずつ解釈しながら実行する（Ｓ２８）。メソッド３の圧縮バイトコードを伸張し、伸張済みフラグをオンにする（Ｓ３０）。メソッド３のバイトコードを１命令ずつ解釈しながら実行する（Ｓ３２）。メソッド１の圧縮バイトコードを伸張し、伸張済みフラグをオンにする（Ｓ３４）。メソッド１のバイトコードを１命令ずつ解釈しながら実行する（Ｓ３６）。

【００６６】

メソッド２の圧縮バイトコードを伸張し、伸張済みフラグをオンにする（Ｓ３８）。メソッド２のバイトコードを１命令ずつ解釈しながら実行する（Ｓ４０）。メソッド４の圧縮バイトコードを伸張し、伸張済みフラグをオンにする（Ｓ４

２）。メソッド４のバイトコードを１命令ずつ解釈しながら実行する（Ｓ４４）。２つめのメソッド４は伸張済みであるため、メソッド４のバイトコードを１命令ずつ解釈しながら実行する（Ｓ４６）。３つめのメソッド４は伸張済みであるため、メソッド４のバイトコードを１命令ずつ解釈しながら実行する（Ｓ４８）。

【００６７】

以上説明したように、本実施の形態によると、バイトコードが圧縮されてＲＯＭに記憶されている。このため、ＲＯＭの容量を小さくすることができる。

【００６８】

［実施の形態２］

本実施の形態は、第１の実施の形態と異なり、圧縮するバイトコードの単位がメソッド単位ではなく基本ブロック単位である。基本ブロックとは、その中に分岐命令を含まない一連の命令のことをいう。

【００６９】

本実施の形態に係るＪａｖａ（Ｒ）仮想マシンは、図１を参照して説明した組込み機器を用いて実現される。このため、その詳細な説明はここでは繰返さない。

【００７０】

また、ＲＯＭ４に記憶されている情報は図２を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【００７１】

なお、圧縮バイトコード格納領域１２には、基本ブロック単位でバイトコードが圧縮され、記憶されているものとする。

【００７２】

図７を参照して、ＲＡＭ８は、オペレーティングシステムを格納するオペレーティングシステム格納領域１６と、バーチャルマシン（ＶＭ）モジュールを格納するＶＭモジュール格納領域１８と、データの圧縮するためのプログラムである圧縮モジュールを格納するための圧縮モジュール格納領域２４と、圧縮されたデータを伸張するためのプログラムである伸張モジュールを格納するための伸張モ

ジュール格納領域２６と、バイトコードおよびネイティブコード等を格納するためのコード格納領域２８と、基本ブロックの状態を表わす基本ブロックステータスを格納するための基本ブロックステータス格納領域６０とを含む。基本ブロックステータス格納領域６０は基本ブロック毎に設けられる。

【００７３】

基本ブロックステータス格納領域６０は、基本ブロックが実行された回数を格納する回数格納領域６２と、基本ブロックのサイズを格納するサイズ格納領域６４と、ネイティブコードが存在するか否かを示すネイティブコード有無フラグを格納するネイティブコード有無フラグ格納領域６６と、バイトコードがコンパイルされた時刻または順番を格納するコンパイル時刻（順番）格納領域６８と、圧縮に関する情報を格納する圧縮情報格納領域７０とを含む。

【００７４】

圧縮情報格納領域７０は、基本ブロックが圧縮されているか否かを示す圧縮フラグを格納する圧縮フラグ格納領域７２と、圧縮された基本ブロックが伸張されているか否かを示す伸張済みフラグを格納する伸張済みフラグ格納領域７４と、基本ブロックの圧縮方式を格納する圧縮方式格納領域７６と、圧縮の際の圧縮率を格納する圧縮率格納領域７８と、圧縮された時刻または順番を格納する圧縮時刻（順番）格納領域８０と、圧縮ネイティブコードのサイズを格納する圧縮サイズ格納領域８２とを含む。

【００７５】

図８を参照して、上位のモジュールよりある基本ブロックの呼出しがあった場合には、ＣＰＵ２は、呼出された基本ブロックの伸張済みフラグ格納領域７４に格納されている伸張済みフラグを参照し、呼出された基本ブロックのバイトコードが伸張されており、伸張済みバイトコード格納領域３０に格納されているか否かを調べる（Ｓ５２）。

【００７６】

バイトコードが伸張されていなければ（Ｓ５２でＮＯ）、圧縮バイトコード格納領域１２に格納されたバイトコードが伸張され（Ｓ５４）、伸張後のバイトコードが伸張済みバイトコード格納領域３０に格納される（Ｓ５６）。次に、この

基本ブロックのバイトコードが伸張済みであることを表わすため、伸張済みフラグの値がオンに変更される（Ｓ５８）。

【００７７】

Ｓ５８の処理の後、または基本ブロックのバイトコードが伸張済みバイトコード格納領域３０に格納されている場合には（Ｓ５２でＹＥＳ）、ＣＰＵ２はインタプリタを用いて、伸張済みのバイトコードを１命令ずつ解釈しながら実行する（Ｓ６０）。その後、呼出し側のモジュールへ復帰するための処理を行なう（Ｓ６２）。

【００７８】

なお、伸張済みバイトコード格納領域３０に書込まれていた伸張済みのバイトコードが削除または他のバイトコードの上書き等により失われる場合には、そのバイトコードに対応した基本ブロックの伸張済みフラグがオフに設定される。

【００７９】

以上説明したように、本実施の形態によると、バイトコードが圧縮されてＲＯＭに記憶されている。このため、ＲＯＭの容量を小さくすることができる。

【００８０】

また、伸張済みバイトコード格納領域には、基本ブロックの圧縮バイトコードを伸張したバイトコードが格納される。このため、第１の実施の形態に比べて伸張済みバイトコード格納領域の容量が少なくてすみ、ＲＡＭの容量を小さくすることができる。

【００８１】

［実施の形態３］

本実施の形態は、第１の実施の形態と異なり、圧縮するバイトコードの単位がメソッド単位ではなく命令単位である。

【００８２】

本実施の形態に係るＪａｖａ（Ｒ）仮想マシンは、図１を参照して説明した組込み機器を用いて実現される。このため、その詳細な説明はここでは繰返さない。

【００８３】

また、ＲＯＭ４に記憶されている情報は図２参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【００８４】

なお、圧縮バイトコード格納領域１２には、命令単位でバイトコードが圧縮され、記憶されているものとする。

【００８５】

図９を参照して、ＲＡＭ８は、オペレーティングシステムを格納するオペレーティングシステム格納領域１６と、バーチャルマシン（ＶＭ）モジュールを格納するＶＭモジュール格納領域１８と、データの圧縮するためのプログラムである圧縮モジュールを格納するための圧縮モジュール格納領域２４と、圧縮されたデータを伸張するためのプログラムである伸張モジュールを格納するための伸張モジュール格納領域２６と、バイトコードおよびネイティブコード等を格納するためのコード格納領域２８と、命令の状態を表わす命令ステータスを格納するための命令ステータス格納領域９０とを含む。命令ステータス格納領域９０は命令毎に設けられる。

【００８６】

命令ステータス格納領域９０は、命令が実行された回数を格納する回数格納領域９２と、命令のサイズを格納するサイズ格納領域９４と、ネイティブコードが存在するか否かを示すネイティブコード有無フラグを格納するネイティブコード有無フラグ格納領域９６と、バイトコードがコンパイルされた時刻または順番を格納するコンパイル時刻（順番）格納領域９８と、圧縮に関する情報を格納する圧縮情報格納領域１００とを含む。

【００８７】

圧縮情報格納領域１００は、命令が圧縮されているか否かを示す圧縮フラグを格納する圧縮フラグ格納領域１０２と、圧縮された命令が伸張されているか否かを示す伸張済みフラグを格納する伸張済みフラグ格納領域１０４と、命令の圧縮方式を格納する圧縮方式格納領域１０６と、圧縮の際の圧縮率を格納する圧縮率格納領域１０８と、圧縮された時刻または順番を格納する圧縮時刻（順番）格納領域１１０と、圧縮ネイティブコードのサイズを格納する圧縮サイズ格納領域１

１２とを含む。

【００８８】

図１０を参照して、上位のモジュールよりある命令の呼出しがあった場合には、ＣＰＵ２は、呼出された命令の伸張済みフラグ格納領域１０４に格納されている伸張済みフラグを参照し、呼出された命令のバイトコードが伸張されており、伸張済みバイトコード格納領域３０に格納されているか否かを調べる（Ｓ７２）。

【００８９】

バイトコードが伸張されていなければ（Ｓ７２でＮＯ）、圧縮バイトコード格納領域１２に格納されたバイトコードが伸張され（Ｓ７４）、伸張後のバイトコードが伸張済みバイトコード格納領域３０に格納される（Ｓ７６）。次に、この命令のバイトコードが伸張済みであることを表わすため、伸張済みフラグの値がオンに変更される（Ｓ７８）。

【００９０】

Ｓ７８の処理の後、または命令のバイトコードが伸張済みバイトコード格納領域３０に格納されている場合には（Ｓ７２でＹＥＳ）、ＣＰＵ２はインタプリタを用いて、伸張済みのバイトコードを解釈し、実行する（Ｓ８０）。その後、呼出し側のモジュールへ復帰するための処理を行なう（Ｓ８２）。

【００９１】

なお、伸張済みバイトコード格納領域３０に書込まれていた伸張済みのバイトコードが削除または他のバイトコードの上書き等により失われる場合には、そのバイトコードに対応した命令の伸張済みフラグがオフに設定される。

【００９２】

以上説明したように、本実施の形態によると、バイトコードが圧縮されてＲＯＭに記憶されている。このため、ＲＯＭの容量を小さくすることができる。

【００９３】

また、伸張済みバイトコード格納領域には、命令の圧縮バイトコードを伸張したバイトコードが格納される。このため、第１および第２の実施の形態に比べて伸張済みバイトコード格納領域の容量が少なくてすみ、ＲＡＭの容量を小さくす

ることができる。

【００９４】

［実施の形態４］

本実施の形態では、圧縮バイトコード格納領域に格納されている圧縮バイトコードの圧縮方法として、複数の圧縮方法がサポートされている。

【００９５】

本実施の形態に係るＪａｖａ（Ｒ）仮想マシンは、図１を参照して説明した組込み機器を用いて実現される。このため、その詳細な説明はここでは繰返さない。

【００９６】

また、ＲＯＭ４およびＲＡＭ８に記憶されている情報も図２および図３をそれぞれ参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【００９７】

なお、圧縮バイトコード格納領域１２には、メソッド単位でバイトコードが圧縮され、記憶されているものとする。

【００９８】

図１１を参照して、上位のモジュールよりあるメソッドの呼出しがあった場合には、ＣＰＵ２は、呼出されたメソッドの伸張済みフラグ格納領域５０に格納されている伸張済みフラグを参照し、呼出されたメソッドのバイトコードが、伸張されており、伸張済みバイトコード格納領域３０に格納されているか否かを調べる（Ｓ９２）。

【００９９】

バイトコードが格納されていなければ（Ｓ９２でＮＯ）、圧縮バイトコード格納領域１２に格納されたバイトコードの圧縮方式が判定される（Ｓ９４）。圧縮方式の判定は圧縮バイトコードのヘッダ部分を参照することにより行なわれる。圧縮バイトコード格納領域１２に格納されたバイトコードがＳ９４で求められた圧縮方式に基づいて伸張され（Ｓ９６）、伸張後のバイトコードが伸張済みバイトコード格納領域３０格納される（Ｓ９８）。次に、このメソッドのバイトコー

ドが伸張済みであることを表わすため、伸張済みフラグの値がオンに変更される（Ｓ１００）。

【０１００】

Ｓ１００の処理の後、またはメソッドのバイトコードが伸張済みバイトコード格納領域３０に格納されている場合には（Ｓ９２でＹＥＳ）、ＣＰＵ２はインタプリタを用いて、伸張済みのバイトコードを１命令ずつ解釈しながら実行する（Ｓ１０２）。その後、呼出し側のモジュールへの復帰をするための処理を行なう（Ｓ１０４）。

【０１０１】

なお、伸張済みバイトコード格納領域３０に書込まれていた伸張済みのバイトコードが削除または他のバイトコードの上書き等により失われる場合には、そのバイトコードに対応したメソッドの伸張済みフラグがオフに設定される。

【０１０２】

以上説明したように、本実施の形態によると、バイトコードが圧縮されてＲＯＭに記憶されている。このため、ＲＯＭの容量を小さくすることができる。

【０１０３】

また、メソッド毎に圧縮方式をユーザが選択することができ、最適な圧縮方式を採用することができるため、ＲＯＭの容量を小さくすることができる。

【０１０４】

［実施の形態５］

本実施の形態ではＪＩＴを用いて、圧縮バイトコードを伸張後、ネイティブコードにコンパイルし、実行するものである。

【０１０５】

本実施の形態に係るＪａｖａ（Ｒ）仮想マシンは、図１を参照して説明した組込み機器を用いて実現される。このため、その詳細な説明はここでは繰返さない。

【０１０６】

また、ＲＯＭ４およびＲＡＭ８に記憶されている情報も図２および図３をそれぞれ参照して説明したものと同様である。このため、その詳細な説明はここでは

繰返さない。

【０１０７】

なお、圧縮バイトコード格納領域１２には、メソッド単位でバイトコードが圧縮され、記憶されているものとする。

【０１０８】

図１２を参照して、上位のモジュールよりあるメソッドの呼出しがあった場合には、ＣＰＵ２は、呼出されたメソッドの伸張済みフラグ格納領域５０に格納されている伸張済みフラグを参照し、呼出されたメソッドのバイトコードが回答されており、かつネイティブコードに変換され、ネイティブコード格納領域３２に格納されているか否かを調べる（Ｓ１１２）。

【０１０９】

バイトコードが伸張されていなければ（Ｓ１１２でＮＯ）、圧縮バイトコード格納領域１２に格納されたバイトコードが伸張され（Ｓ１１４）、伸張後のバイトコードがＪＩＴ格納領域２２に格納されたＪＩＴによりコンパイルされ、ネイティブコードに変換される（Ｓ１１６）。ネイティブコードは、ネイティブコード格納領域３２に格納される（Ｓ１１８）。このメソッドの圧縮バイトコードが伸張済みおよびコンパイル済みであることを表わすため、伸張済みフラグの値がオンに設定される（Ｓ１２０）。

【０１１０】

Ｓ１２０の処理の後、またはメソッドのネイティブコードがネイティブコード格納領域３２に格納されている場合には（Ｓ１１２でＹＥＳ）、ＣＰＵ２は、プログラムカウンタの値をネイティブコードの先頭アドレスに設定し、ネイティブコードを実行する（Ｓ１２２）。その後、呼出し側のモジュールへ復帰するための処理を行なう（Ｓ１２４）。

【０１１１】

なお、ネイティブコード格納領域３２に格納されていたネイティブコードが削除または他のネイティブコードの上書き等により失われる場合には、そのネイティブコードに対応したメソッドの伸張済みフラグがオフに設定される。

【０１１２】

図４を参照して、Ｊａｖａ（Ｒ）言語で記述されたプログラムが、メソッド１～４の４つのメソッドにより構成されるものとし、メソッド１、メソッド２、メソッド３、メソッド１、メソッド２、メソッド４、メソッド４、メソッド４の順に呼出されて、実行されるものとする。なお、ネイティブコード格納領域３２には、４つのメソッドのネイティブコードを一度にすべて格納することができるものとする。このとき、図１３を参照して、ＣＰＵ２は以下の順序で処理を実行する。

【０１１３】

メソッド１の圧縮バイトコードを伸張し、伸張後のバイトコードをネイティブコードに変換するとともに、伸張済みフラグをオンにする（Ｓ１３２）。メソッド１のネイティブコードが１命令ずつ実行される（Ｓ１３４）。

【０１１４】

メソッド２の圧縮バイトコードを伸張し、伸張後のバイトコードをネイティブコードに変換するとともに、伸張済みフラグをオンにする（Ｓ１３６）。メソッド２のネイティブコードが１命令ずつ実行される（Ｓ１３８）。

【０１１５】

メソッド３の圧縮バイトコードを伸張し、伸張後のバイトコードをネイティブコードに変換するとともに、伸張済みフラグをオンにする（Ｓ１４０）。メソッド３のネイティブコードが１命令ずつ実行される（Ｓ１４２）。

【０１１６】

メソッド１のネイティブコードが１命令ずつ実行される（Ｓ１４４）。メソッド２のネイティブコードが１命令ずつ実行される（Ｓ１４６）。

【０１１７】

メソッド４の圧縮バイトコードを伸張し、伸張後のバイトコードをネイティブコードに変換するとともに、伸張済みフラグをオンにする（Ｓ１４８）。メソッド４のネイティブコードが１命令ずつ実行される（Ｓ１５０）。メソッド４のネイティブコードが１命令ずつ実行される（Ｓ１５２）。メソッド４のネイティブコードが１命令ずつ実行される（Ｓ１５４）。

【０１１８】

以上説明したように、本実施の形態によると、バイトコードが圧縮されてＲＯＭに記憶されている。このため、ＲＯＭの容量を小さくすることができる。

【０１１９】

なお、ネイティブコード格納領域３２をＣＰＵ２とＲＡＭ８との間に設けられたキャッシュメモリ（図示せず）に設けることも可能である。このようにすれば、処理の実行速度を向上させることができる。

【０１２０】

［実施の形態６］

本実施の形態では、ＪＩＴを用いてバイトコードを、ネイティブコードにコンパイルし、実行するものである。その際、不要になったネイティブコードを圧縮して保存しておく。

【０１２１】

本実施の形態に係るＪａｖａ（Ｒ）仮想マシンは、図１を参照して説明した組込み機器を用いて実現される。このため、その詳細な説明はここでは繰返さない。

【０１２２】

また、ＲＡＭ８に記憶されている情報は図３を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０１２３】

図１４を参照して、ＲＯＭ４は、Ｊａｖａ（Ｒ）言語で記述されたプログラムのバイトコードを格納するバイトコード格納領域１２２と、その他のクラス情報を格納する領域１２４とを含む。

【０１２４】

図１５を参照して、上位のモジュールよりあるメソッドの呼出しがあった場合には、ＣＰＵ２は、ネイティブコード有無フラグ格納領域４２に格納されているそのメソッドのネイティブコード有無フラグを参照し、ネイティブコード格納領域３２に呼出されたネイティブコードが格納されているか否かを判断する（Ｓ１６２）。

【０１２５】

ネイティブコード格納領域３２にネイティブコードが格納されていなければ（Ｓ１６２でＮＯ）、ネイティブコード格納領域３２にネイティブコードを追加格納するだけの空きがあるか否か判断する（Ｓ１６４）。

【０１２６】

ネイティブコード格納領域３２に空きがなければ（Ｓ１６４でＮＯ）、現在、ネイティブコード格納領域３２に存在するメソッドのネイティブコード有無フラグをオフにし、ネイティブコード格納領域３２に空きを作る（Ｓ１６６）。

【０１２７】

ネイティブコード格納領域３２に空きがあるか（Ｓ１６４でＹＥＳ）、ネイティブコード格納領域３２に空きを作った場合には（Ｓ１６６）、圧縮ネイティブコード格納領域３４に呼出されたメソッドの圧縮ネイティブコードが格納されているか否かを調べる（Ｓ１６８）。すなわち、そのメソッドの圧縮フラグ格納領域４８に格納された圧縮フラグがオンになっているか否かを調べる。

【０１２８】

圧縮ネイティブコード格納領域３４に呼出されたメソッドの圧縮ネイティブコードが格納されていれば（Ｓ１６８でＹＥＳ）、圧縮ネイティブコードを伸張した後（Ｓ１７０）、ネイティブコード格納領域３２に格納する（Ｓ１７４）。

【０１２９】

圧縮ネイティブコード格納領域３４に呼出されたメソッドの圧縮ネイティブコードが格納されていなければ、そのメソッドのバイトコードをＲＯＭ４のバイトコード格納領域１２２より読出し、ネイティブコードに変換した後（Ｓ１７２）、ネイティブコード格納領域３２に格納する（Ｓ１７４）。

【０１３０】

Ｓ１７４の処理の後、呼出されたメソッドのネイティブコード有無フラグをオンする。Ｓ１７６の処理の後、またはネイティブコード格納領域３２に呼出されたメソッドのネイティブコードが格納されている場合には（Ｓ１６２でＹＥＳ）、ＣＰＵ２は、ネイティブコードを実行する（Ｓ１７８）。

【０１３１】

呼出されたメソッドの圧縮ネイティブコードが圧縮ネイティブコード格納領域

３４に格納されているか否か判断される（Ｓ１８０）。圧縮ネイティブコードが圧縮ネイティブコード格納領域３４に格納されていなければ、ネイティブコード格納領域３２に格納されている、呼出されたメソッドのネイティブコードを圧縮し、圧縮ネイティブコード格納領域３４に格納する（Ｓ１８２）。また、圧縮フラグ格納領域４８に格納されているこのメソッドの圧縮フラグをオンにする（Ｓ１８４）。

【０１３２】

圧縮ネイティブコードが圧縮ネイティブコード格納領域３４に格納されているか（Ｓ１８０でＹＥＳ）、圧縮ネイティブコード格納領域３４に格納した場合には（Ｓ１８４）、呼出し側のモジュールへ復帰するための処理を行なう（Ｓ１８６）。

【０１３３】

図４を参照して、Ｊａｖａ（Ｒ）言語で記述されたプログラムが、メソッド１～４の４つのメソッドにより構成されるものとし、メソッド１、メソッド２、メソッド３、メソッド１、メソッド２、メソッド４、メソッド４、メソッド４の順に呼出されて、実行されるものとする。

【０１３４】

なお、ネイティブコード格納領域３２には、１つのメソッドのネイティブコードしか格納することができないものとし、圧縮ネイティブコード格納領域３４にはすべてのメソッドの圧縮ネイティブコードを格納できるものとする。

【０１３５】

このとき、図１６および図１７を参照して、ＣＰＵ２は以下の順序で処理を実行する。

【０１３６】

メソッド１をネイティブコード１に変換し、ネイティブコード格納領域３２に格納する（Ｓ１９２）。ネイティブコード１を実行する（Ｓ１９４）。ネイティブコード１を圧縮し、圧縮ネイティブコード格納領域３４に格納する（Ｓ１９６）。

【０１３７】

メソッド２をネイティブコード２に変換し、ネイティブコード格納領域３２に格納する（Ｓ１９８）。ネイティブコード２を実行する（Ｓ２００）。ネイティブコード２を圧縮し、圧縮ネイティブコード格納領域３４に格納する（Ｓ２０２）。

【０１３８】

メソッド３をネイティブコード３に変換し、ネイティブコード格納領域３２に格納する（Ｓ２０４）。ネイティブコード３を実行する（Ｓ２０６）。ネイティブコード３を圧縮し、圧縮ネイティブコード格納領域３４に格納する（Ｓ２０８）。

【０１３９】

圧縮ネイティブコード格納領域３４に格納されているメソッド１の圧縮ネイティブコードを伸張し、実行する（Ｓ２１０）。圧縮ネイティブコード格納領域３４に格納されているメソッド２の圧縮ネイティブコードを伸張し、実行する（Ｓ２１２）。

【０１４０】

メソッド４をネイティブコード４に変換し、ネイティブコード格納領域３２に格納する（Ｓ２１４）。ネイティブコード４を実行する（Ｓ２１６）。ネイティブコード４を圧縮し、圧縮ネイティブコード格納領域３４に格納する（Ｓ２１８）。ネイティブコード格納領域３２に格納されているネイティブコード４を２回連続して実行する（Ｓ２２０）。

【０１４１】

以上説明したように、本実施の形態によると、バイトコードがネイティブコードに変換された後、圧縮されて記憶される。このため、ＪＩＴを用いた従来のＪａｖａ（Ｒ）ＶＭに比べ、ＲＡＭの使用容量を削減することが可能になる。

【０１４２】

［実施の形態７］

本実施の形態では、第６の実施の形態と異なり、ネイティブコードを圧縮するための圧縮方式として複数種類の圧縮方式が用意されているものとする。

【０１４３】

本実施の形態に係るＪａｖａ（Ｒ）仮想マシンは、図１を参照して説明した組込み機器を用いて実現される。このため、その詳細な説明はここでは繰返さない。

【０１４４】

また、ＲＡＭ８に記憶されている情報は図３を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０１４５】

さらに、ＲＯＭ４に記憶されている情報は図１４を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０１４６】

図１８を参照して、上位モジュールよりあるメソッドの呼出しがあった場合のＣＰＵ２の行う処理について説明する。ＣＰＵ２の行なう処理は、図１５を参照して説明した処理においてＳ１７０の処理の代わりにＳ２２２の処理を実行し、Ｓ１８２およびＳ１８４の処理の代わりに、Ｓ２２４～Ｓ２２８の処理を実行するものである。

【０１４７】

Ｓ２２２では、呼出されたメソッドの伸張を行なうが、どのような方式で圧縮されているかを圧縮方式格納領域５２に格納された値から調べ、その方式に基づいたメソッドのネイティブコードの伸張を行なう。

【０１４８】

Ｓ２２４では、所定の規則に基づいて、呼出されたメソッドのネイティブコードの圧縮に最適な方式を判定する。Ｓ２２６では、選択された圧縮方式でネイティブコードを圧縮し、圧縮ネイティブコード格納領域３４に格納する。Ｓ２２８では、圧縮フラグ格納領域４８に格納されているこのメソッドの圧縮フラグをオンにするとともに、圧縮方式を圧縮方式格納領域５２に格納する。

【０１４９】

図４を参照して、Ｊａｖａ（Ｒ）言語で記述されたプログラムが、メソッド１～４の４つのメソッドにより構成されるものとし、メソッド１、メソッド２、メソッド３、メソッド１、メソッド２、メソッド４、メソッド４、メソッド４の順

に呼出されて、実行されるものとする。

【０１５０】

なお、ネイティブコード格納領域３２には、１つのメソッドのネイティブコードしか格納することができないものとし、圧縮ネイティブコード格納領域３４にはすべてのメソッドの圧縮ネイティブコードを格納できるものとする。

【０１５１】

また、圧縮方式には圧縮方式ＡおよびＢの２種類があるものとし、メソッド１のネイティブコードの圧縮には圧縮方式Ａが適しており、メソッド２～４のネイティブコードの圧縮には圧縮方式Ｂが適しているものとする。

【０１５２】

このとき、図１９、図２０および図２１を参照して、ＣＰＵ２は以下の順序で処理を実行する。

【０１５３】

メソッド１をネイティブコード１に変換し、ネイティブコード格納領域３２に格納する（Ｓ２３２）。ネイティブコード１を実行する（Ｓ２３４）。ネイティブコード１を圧縮するのに最適な圧縮方式Ａを選択し、ネイティブコード１を圧縮した後、圧縮ネイティブコード格納領域３４に格納する（Ｓ２３６）。ネイティブコード１が圧縮方式Ａで圧縮されたことを圧縮方式格納領域５２に格納する（Ｓ２３８）。

【０１５４】

メソッド２をネイティブコード２に変換し、ネイティブコード格納領域３２に格納する（Ｓ２４０）。ネイティブコード２を実行する（Ｓ２４２）。ネイティブコード２を圧縮するのに最適な圧縮方式Ｂを選択し、ネイティブコード２を圧縮した後、圧縮ネイティブコード格納領域３４に格納する（Ｓ２４４）。ネイティブコード２が圧縮方式Ｂで圧縮されたことを圧縮方式格納領域５２に格納する（Ｓ２４６）。

【０１５５】

メソッド３をネイティブコード３に変換し、ネイティブコード格納領域３２に格納する（Ｓ２４８）。ネイティブコード３を実行する（Ｓ２５０）。ネイティ

ブコード３を圧縮するのに最適な圧縮方式Ｂを選択し、ネイティブコード３を圧縮した後、圧縮ネイティブコード格納領域３４に格納する（Ｓ２５２）。ネイティブコード３が圧縮方式Ｂで圧縮されたことを圧縮方式格納領域５２に格納する（Ｓ２５４）。

【０１５６】

圧縮ネイティブコード格納領域３４に格納されているメソッド１の圧縮ネイティブコードの圧縮方式を圧縮方式格納領域５２に記憶されている値より調べる（Ｓ２５６）。その結果、圧縮方式は、圧縮方式Ａであることがわかる。圧縮ネイティブコード格納領域３４に格納されているメソッド１の圧縮ネイティブコードを圧縮方式Ａに対応した方式で伸張し、実行する（Ｓ２５８）。

【０１５７】

圧縮ネイティブコード格納領域３４に格納されているメソッド２の圧縮ネイティブコードの圧縮方式を圧縮方式格納領域５２に記憶されている値より調べる（Ｓ２６０）。その結果、圧縮方式は、圧縮方式Ｂであることがわかる。圧縮ネイティブコード格納領域３４に格納されているメソッド２の圧縮ネイティブコードを圧縮方式Ｂに対応した方式で伸張し、実行する（Ｓ２６２）。

【０１５８】

メソッド４をネイティブコード４に変換し、ネイティブコード格納領域３２に格納する（Ｓ２６４）。ネイティブコード４を実行する（Ｓ２６６）。ネイティブコード４を圧縮するのに最適な圧縮方式Ｂを選択し、ネイティブコード４を圧縮した後、圧縮ネイティブコード格納領域３４に格納する（Ｓ２６８）。ネイティブコード４が圧縮方式Ｂで圧縮されたことを圧縮方式格納領域５２に格納する（Ｓ２７０）。ネイティブコード格納領域３２に格納されているネイティブコード４を２回連続して実行する（Ｓ２７２）。

【０１５９】

以上説明したように、本実施の形態によると、バイトコードがネイティブコードに変換された後、圧縮されて記憶される。このため、ＪＩＴを用いた従来のＪａｖａ（Ｒ）ＶＭに比べ、ＲＡＭの使用容量を削減することが可能になる。

【０１６０】

また、メソッド毎に最適な圧縮方式を用いてネイティブコードの圧縮が行なわれる。このため、圧縮ネイティブコード格納領域３４の容量を第６の実施の形態よりも小さくすることができる。

【０１６１】

［実施の形態８］

本実施の形態では、第６および第７の実施の形態と異なり、ネイティブコード格納領域３２および圧縮ネイティブコード格納領域３４の格納できるメソッドの数に制限が設けられているものとする。

【０１６２】

本実施の形態に係るＪａｖａ（Ｒ）仮想マシンは、図１を参照して説明した組込み機器を用いて実現される。このため、その詳細な説明はここでは繰返さない。

【０１６３】

また、ＲＡＭ８に記憶されている情報は図３を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０１６４】

さらに、ＲＯＭ４に記憶されている情報は図１４を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０１６５】

図２２および図２３を参照して、上位モジュールよりあるメソッドＮの呼出しがあった場合のＣＰＵ２の行う処理について説明する。ＣＰＵ２は、ネイティブコード有無フラグ格納領域４２に格納されているメソッドＮのネイティブコード有無フラグを参照し、ネイティブコード格納領域３２にメソッドＮのネイティブコードが格納されているか否かを判断する（Ｓ２８２）。

【０１６６】

ネイティブコード格納領域３２にメソッドＮのネイティブコードが格納されていなければ（Ｓ２８２でＮＯ）、ネイティブコード格納領域３２にネイティブコードを追加格納するだけの空きがあるか否か判断する（Ｓ２８４）。

【０１６７】

ネイティブコード格納領域３２に空きがなければ（Ｓ２８４でＮＯ）、コンパイル時刻（順番）格納領域４４の値を調べ、ネイティブコード格納領域３２内で最も古いメソッドを調べ、そのメソッドをメソッドＡとする（Ｓ２８６）。

【０１６８】

メソッドＡの圧縮ネイティブコードが圧縮ネイティブコード格納領域３４に格納されているか否かを判断する（Ｓ２８８）。メソッドＡの圧縮ネイティブコードが圧縮ネイティブコード格納領域３４に格納されていなければ（Ｓ２８８でＮＯ）、圧縮ネイティブコード格納領域３４の空きが十分か否かを調べる（Ｓ２９０）。圧縮ネイティブコード格納領域３４の空きが十分でなければ（Ｓ２９０でＮＯ）、回数格納領域３８に格納された回数に基づいて、圧縮ネイティブコード格納領域３４内で最も実行頻度の低いメソッドを調べ、そのメソッドをメソッドＢとする（Ｓ２９２）。メソッドＢの圧縮ネイティブコードを解放、すなわち削除し（Ｓ２９４）、圧縮フラグ格納領域４８に格納されたメソッドＢの圧縮フラグをオフにする（Ｓ２９６）。その後、Ｓ２９０に戻る。

【０１６９】

圧縮ネイティブコード格納領域３４の空きが十分であれば（Ｓ２９０でＹＥＳ）、メソッドＡのネイティブコードを圧縮し、圧縮ネイティブコード格納領域３４に格納する（Ｓ２９８）。また、圧縮フラグ格納領域４８に格納されているメソッドＡの圧縮フラグをオンにする（Ｓ３００）。

【０１７０】

メソッドＡの圧縮ネイティブコードが圧縮ネイティブコード格納領域３４に格納されている場合（Ｓ２８８でＹＥＳ）、またはＳ３００の処理の後、ネイティブコード格納領域３２のうち、メソッドＡのネイティブコードが格納されている領域を解放し（Ｓ３０２）、ネイティブコード有無フラグ格納領域４２に格納されたメソッドＡのネイティブコード有無フラグをオフにする（Ｓ３０４）。その後、Ｓ２８４に戻る。

【０１７１】

ネイティブコード格納領域３２にメソッドＮのネイティブコードを追加格納するだけの空きがあれば（Ｓ２８４でＹＥＳ）、メソッドＮのバイトコードをネイ

ティブコードに変換した後（Ｓ３０６）、ネイティブコード格納領域３２に格納する（Ｓ３０８）。その後、メソッドＮのネイティブコード有無フラグをオンにし、メソッドＮがコンパイルされた順番または時刻をコンパイル時刻（順番）格納領域４４に記憶する（Ｓ３１０）。

【０１７２】

ネイティブコード格納領域３２にメソッドＮのネイティブコードが格納されている場合（Ｓ２８２でＹＥＳ）、またはＳ３１０の処理の後、ＣＰＵ２は、メソッドＮのネイティブコードを実行する（Ｓ３１２）。また、回数格納領域３８に格納されたメソッドＮの実行回数を１つインクリメントする（Ｓ３１４）。その後、メソッドＮの呼出し側へ戻るための処理を行なう（Ｓ３１６）。

【０１７３】

図２４を参照して、Ｊａｖａ（Ｒ）言語で記述されたプログラムがメソッド１～７の７つのメソッドにより構成されるものとし、メソッド１、メソッド２、メソッド３、メソッド１、メソッド２、メソッド４、メソッド５、メソッド６、メソッド７の順に呼出されて、実行されるものとする。

【０１７４】

なお、ネイティブコード格納領域３２に格納できるメソッドの数は３つであり、圧縮ネイティブコード格納領域３４に格納できるメソッドの数は３つであるものとする。

【０１７５】

このとき、図２５、図２６および図２７を参照して、ＣＰＵ２は以下の順序で処理を実行する。

【０１７６】

メソッド１をネイティブコード１に変換し、ネイティブコード格納領域３２に格納する（Ｓ３２２）。ネイティブコード１を実行する（Ｓ３２４）。メソッド２をネイティブコード２に変換し、ネイティブコード格納領域３２に格納する（Ｓ３２６）。ネイティブコード２を実行する（Ｓ３２８）。メソッド３をネイティブコード３に変換し、ネイティブコード格納領域３２に格納する（Ｓ３３０）。メソッド３を実行する（Ｓ３３２）。この時点で、ネイティブコード格納領域

３２には３つのメソッドのネイティブコードが記憶されている。このため、ネイティブコード１～３以外のネイティブコードを格納するためには、いずれかのネイティブコードが確保している領域を解放しなければならない。

【０１７７】

ＣＰＵ２は、ネイティブコード１を実行し（Ｓ３３４）、ネイティブコード２を実行する（Ｓ３３６）。

【０１７８】

メソッド４を実行するに際し、ＣＰＵ２は、ネイティブコード格納領域３２に格納されているネイティブコードの中で最も古いネイティブコード１を圧縮し、圧縮ネイティブコード格納領域３４に格納する。それとともに、ネイティブコード格納領域３２のネイティブコード１が格納されていた領域を解放する（Ｓ３３８）。メソッド４をネイティブコードに変換し、ネイティブコード格納領域３２のネイティブコード１が格納されていた領域に格納する（Ｓ３４０）。ネイティブコード４を実行する（Ｓ３４２）。

【０１７９】

メソッド５を実行するに際し、ＣＰＵ２は、ネイティブコード格納領域３２に格納されているネイティブコードの中で最も古いネイティブコード２を圧縮し、圧縮ネイティブコード格納領域３４に格納する。それとともに、ネイティブコード格納領域３２のネイティブコード２が格納されていた領域を解放する（Ｓ３４４）。メソッド５をネイティブコードに変換し、ネイティブコード格納領域３２のネイティブコード２が格納されていた領域に格納する（Ｓ３４６）。ネイティブコード５を実行する（Ｓ３４８）。

【０１８０】

メソッド６を実行するに際し、ＣＰＵ２は、ネイティブコード格納領域３２に格納されているネイティブコードの中で最も古いネイティブコード３を圧縮し、圧縮ネイティブコード格納領域３４に格納する。それとともに、ネイティブコード格納領域３２のネイティブコード３が格納されていた領域を解放する（Ｓ３５０）。メソッド６をネイティブコードに変換し、ネイティブコード格納領域３２のネイティブコード３が格納されていた領域に格納する（Ｓ３４６）。ネイティ

ブコード６を実行する（Ｓ３４８）。

【０１８１】

図２７を参照して、ここまでの時点でネイティブコード格納領域３２には、ネイティブコード４～７が格納されており、圧縮ネイティブコード格納領域３４には、ネイティブコード１～３が格納されている。

【０１８２】

図２６を参照して、ＣＰＵ２は、メソッド７のバイトコードをネイティブコードに変換する前にネイティブコード格納領域３２に空き領域があるか否かを調べる（Ｓ３５６）。ネイティブコード格納領域３２には空き領域がないため、ネイティブコード格納領域３２に格納されているネイティブコードのうち、最も古いネイティブコード４を圧縮対象とする（Ｓ３５８）。

【０１８３】

ＣＰＵ２は、ネイティブコード４を圧縮する前に、圧縮ネイティブコード格納領域３４に空きがあるか否かを調べる（Ｓ３６０）。圧縮ネイティブコード格納領域３４に空きがないため、圧縮ネイティブコード格納領域３４に格納されている圧縮ネイティブコードのうち実行頻度の最も低い圧縮ネイティブコード３の領域を解放する（Ｓ３６２）。ＣＰＵ２は、ネイティブコード４を圧縮し、圧縮ネイティブコード格納領域３４の圧縮ネイティブコード３が格納されていた領域に格納する。それとともに、ネイティブコード格納領域３２のネイティブコード４が格納されていた領域を解放する（Ｓ３６４）。

【０１８４】

ＣＰＵ２は、メソッド７のバイトコードをネイティブコード７に変換し、ネイティブコード格納領域３２のメソッド７が格納されていた領域に格納する（Ｓ３６６）。ＣＰＵ２は、ネイティブコード７を実行する（Ｓ３６８）。

【０１８５】

以上説明したように、本実施の形態によると、バイトコードがネイティブコードに変換された後、圧縮されて記憶される。このため、ＪＩＴを用いた従来のＪａｖａ（Ｒ）ＶＭに比べ、ＲＡＭの使用容量を削減することが可能になる。

【０１８６】

また、ネイティブコード格納領域３２に複数のメソッドのネイティブコードを格納することができるため、第７の実施の形態に比べて、処理が高速である。

【０１８７】

［実施の形態９］

本実施の形態は、第８の実施の形態と異なり、ネイティブコード格納領域３２に格納しきれなくなったネイティブコードを圧縮する際のネイティブコードの選択基準として、最も実行頻度の低いネイティブコードを選択するものである。

【０１８８】

本実施の形態に係るＪａｖａ（Ｒ）仮想マシンは、図１を参照して説明した組込み機器を用いて実現される。このため、その詳細な説明はここでは繰返さない。

【０１８９】

また、ＲＡＭ８に記憶されている情報は図３を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０１９０】

さらに、ＲＯＭ４に記憶されている情報は図１４を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０１９１】

図２８および図２９を参照して、上位モジュールよりメソッドＮの呼出しがあった場合のＣＰＵ２の行なう処理について説明する。ＣＰＵ２の行なう処理は、図２２のＳ２８６の代わりに、図２８のＳ３７２の処理を実行し、図２３のＳ３１０の処理の代わりに、図２９のＳ３７４の処理を実行するものである。その他の処理は、図２２および図２３を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０１９２】

図２８のＳ３７２では、回数格納領域３８に格納されたネイティブコードの実行回数を調べ、ネイティブコード格納領域３２内で最も実行頻度の低いメソッドを調べ、そのメソッドＡとする。

【０１９３】

図２９のＳ３７４では、メソッドＮのネイティブコード有無フラグをオンにする。

【０１９４】

以上説明したように、本実施の形態によると、バイトコードがネイティブコードに変換された後、圧縮されて記憶される。このため、ＪＩＴを用いた従来のＪａｖａ（Ｒ）ＶＭに比べ、ＲＡＭの使用容量を削減することが可能になる。

【０１９５】

また、ネイティブコード格納領域３２に複数のメソッドのネイティブコードを格納することができるため、第７の実施の形態に比べて、処理が高速である。

【０１９６】

［実施の形態１０］

本実施の形態は、第８の実施の形態と異なり、ネイティブコード格納領域３２に格納しきれなくなったネイティブコードを圧縮する際のネイティブコードの選択基準として、最もサイズの大きいネイティブコードを選択するものである。

【０１９７】

本実施の形態に係るＪａｖａ（Ｒ）仮想マシンは、図１を参照して説明した組込み機器を用いて実現される。このため、その詳細な説明はここでは繰返さない。

【０１９８】

また、ＲＡＭ８に記憶されている情報は図３を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０１９９】

さらに、ＲＯＭ４に記憶されている情報は図１４を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０２００】

図３０および図３１を参照して、上位モジュールよりメソッドＮの呼出しがあった場合のＣＰＵ２の行なう処理について説明する。ＣＰＵ２の行なう処理は、図２２のＳ２８６の代わりに、図３０のＳ３８２の処理を実行し、図２３のＳ３１０の処理の代わりに、図３１のＳ３８４の処理を実行するものである。その他

の処理は、図２２および図２３を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０２０１】

図３０のＳ３８２では、サイズ格納領域４０に格納されたネイティブコードのサイズを調べることにより、ネイティブコード格納領域３２内で最も大きいサイズのネイティブコードを有するメソッドを調べ、メソッドＡとする。

【０２０２】

図３１のＳ３８４では、メソッドＮのネイティブコード有無フラグをオンにし、メソッドＮのネイティブコードのサイズをサイズ格納領域４０に格納する。

【０２０３】

以上説明したように、本実施の形態によると、バイトコードがネイティブコードに変換された後、圧縮されて記憶される。このため、ＪＩＴを用いた従来のＪａｖａ（Ｒ）ＶＭに比べ、ＲＡＭの使用容量を削減することが可能になる。

【０２０４】

また、ネイティブコード格納領域３２に複数のメソッドのネイティブコードを格納することができるため、第７の実施の形態に比べて、処理が高速である。

【０２０５】

［実施の形態１１］

本実施の形態は、第８の実施の形態と異なり、ネイティブコード格納領域３２に格納しきれなくなったネイティブコードを圧縮する際のネイティブコードの選択基準として、最も圧縮率の大きいネイティブコードを選択するものである。

【０２０６】

本実施の形態に係るＪａｖａ（Ｒ）仮想マシンは、図１を参照して説明した組込み機器を用いて実現される。このため、その詳細な説明はここでは繰返さない。

【０２０７】

また、ＲＡＭ８に記憶されている情報は図３を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０２０８】

さらに、ＲＯＭ４に記憶されている情報は図１４を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０２０９】

図３２および図３３を参照して、上位モジュールよりメソッドＮの呼出しがあった場合のＣＰＵ２の行なう処理について説明する。ＣＰＵ２の行なう処理は、図２２のＳ２８６の代わりに、図３２のＳ３９２の処理を実行し、図２３のＳ３１０の処理の代わりに、図３３のＳ３９４の処理を実行するものである。その他の処理は、図２２および図２３を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０２１０】

図３２のＳ３９２では、圧縮率格納領域５４に格納されたネイティブコードを圧縮した際の圧縮率を調べることにより、ネイティブコード格納領域３２内で最も圧縮率の大きいネイティブコードを有するメソッドを調べ、メソッドＡとする。

【０２１１】

図３３のＳ３９４では、メソッドＮのネイティブコード有無フラグをオンにし、メソッドＮのネイティブコードを圧縮した際の圧縮率を圧縮率格納領域５４に格納する。

【０２１２】

以上説明したように、本実施の形態によると、バイトコードがネイティブコードに変換された後、圧縮されて記憶される。このため、ＪＩＴを用いた従来のＪａｖａ（Ｒ）ＶＭに比べ、ＲＡＭの使用容量を削減することが可能になる。

【０２１３】

また、ネイティブコード格納領域３２に複数のメソッドのネイティブコードを格納することができるため、第７の実施の形態に比べて、処理が高速である。

【０２１４】

［実施の形態１２］

本実施の形態は、第８の実施の形態と異なり、圧縮ネイティブコード格納領域３４に格納しきれなくなった圧縮ネイティブコードを解放する際の圧縮ネイティ

ブコードの選択基準として、最もサイズの大きい圧縮ネイティブコードを選択するものである。

【０２１５】

本実施の形態に係るＪａｖａ（Ｒ）仮想マシンは、図１を参照して説明した組込み機器を用いて実現される。このため、その詳細な説明はここでは繰返さない。

【０２１６】

また、ＲＡＭ８に記憶されている情報は図３を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０２１７】

さらに、ＲＯＭ４に記憶されている情報は図１４を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０２１８】

図３４および図３５を参照して、上位モジュールよりメソッドＮの呼出しがあった場合のＣＰＵ２の行なう処理について説明する。ＣＰＵ２の行なう処理は、図２２のＳ２９２の処理の代わりに、図３４のＳ４０２の処理を実行し、図２３のＳ３１４の処理の代わりに、図３５のＳ４０４の処理を実行するものである。その他の処理は、図２２および図２３を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０２１９】

図３４のＳ４０２では、圧縮サイズ格納領域５８に格納された圧縮ネイティブコードのサイズを調べることにより、圧縮ネイティブコード格納領域３４内で最もサイズの大きい圧縮ネイティブコードを有するメソッドを調べ、メソッドＡとする。

【０２２０】

図３５のＳ４０４では、メソッドＮのネイティブコードを圧縮した際のサイズを調べ、そのサイズを圧縮サイズ格納領域５８に格納する。

【０２２１】

以上説明したように、本実施の形態によると、バイトコードがネイティブコー

ドに変換された後、圧縮されて記憶される。このため、ＪＩＴを用いた従来のＪａｖａ（Ｒ）ＶＭに比べ、ＲＡＭの使用容量を削減することが可能になる。

【０２２２】

また、ネイティブコード格納領域３２に複数のメソッドのネイティブコードを格納することができるため、第７の実施の形態に比べて、処理が高速である。

【０２２３】

［実施の形態１３］

本実施の形態は、第８の実施の形態と異なり、圧縮ネイティブコード格納領域３４に格納しきれなくなった圧縮ネイティブコードを解放する際の圧縮ネイティブコードの選択基準として、圧縮率が最も低い圧縮ネイティブコードを選択するものである。

【０２２４】

本実施の形態に係るＪａｖａ（Ｒ）仮想マシンは、図１を参照して説明した組込み機器を用いて実現される。このため、その詳細な説明はここでは繰返さない。

【０２２５】

また、ＲＡＭ８に記憶されている情報は図３を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０２２６】

さらに、ＲＯＭ４に記憶されている情報は図１４を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０２２７】

図３６および図３７を参照して、上位モジュールよりメソッドＮの呼出しがあった場合のＣＰＵ２の行なう処理について説明する。ＣＰＵ２の行なう処理は、図２２のＳ２９２の処理の代わりに、図３６のＳ４１２の処理を実行し、図２３のＳ３１４の処理の代わりに、図３７のＳ４１４の処理を実行するものである。その他の処理は、図２２および図２３を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０２２８】

図３６のＳ４１２では、圧縮率格納領域５４に格納されたネイティブコードの圧縮率を調べることにより、圧縮ネイティブコード格納領域３４内で最も圧縮率の低い圧縮ネイティブコードを有するメソッドを調べ、メソッドＡとする。

【０２２９】

図３７のＳ４１４では、メソッドＮのネイティブコードを圧縮した際の圧縮率を調べ、圧縮率格納領域５４に格納する。

【０２３０】

以上説明したように、本実施の形態によると、バイトコードがネイティブコードに変換された後、圧縮されて記憶される。このため、ＪＩＴを用いた従来のＪａｖａ（Ｒ）ＶＭに比べ、ＲＡＭの使用容量を削減することが可能になる。

【０２３１】

また、ネイティブコード格納領域３２に複数のメソッドのネイティブコードを格納することができるため、第７の実施の形態に比べて、処理が高速である。

【０２３２】

［実施の形態１４］

本実施の形態は、第８の実施の形態と異なり、圧縮ネイティブコード格納領域３４に格納しきれなくなった圧縮ネイティブコードを解放する際の圧縮ネイティブコードの選択基準として、最も古く圧縮された圧縮ネイティブコードを選択するものである。

【０２３３】

本実施の形態に係るＪａｖａ（Ｒ）仮想マシンは、図１を参照して説明した組込み機器を用いて実現される。このため、その詳細な説明はここでは繰返さない。

【０２３４】

また、ＲＡＭ８に記憶されている情報は図３を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０２３５】

さらに、ＲＯＭ４に記憶されている情報は図１４を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０２３６】

図３８および図３９を参照して、上位モジュールよりメソッドＮの呼出しがあった場合のＣＰＵ２の行なう処理について説明する。ＣＰＵ２の行なう処理は、図２２のＳ２９２およびＳ２９８の処理の代わりに、図３８のＳ４２２およびＳ４２４の処理をそれぞれ行なうものである。また、図２３のＳ３１４の処理を省略したもの（図３９の処理）である。その他の処理は、図２２および図２３を参照して説明したものと同様である。このため、その詳細な説明はここでは繰返さない。

【０２３７】

図３８のＳ４２２では、圧縮時刻（順番）格納領域５６に格納された圧縮時刻を調べることにより、圧縮ネイティブコード格納領域３４内で最初に圧縮された圧縮ネイティブコードを有するメソッドを調べ、メソッドＡとする。

【０２３８】

図３８のＳ４２４では、メソッドＡのネイティブコードを圧縮し、圧縮ネイティブコード格納領域３４に格納する。その際、圧縮した時刻を圧縮時刻（順番）格納領域５６に格納する。

【０２３９】

以上説明したように、本実施の形態によると、バイトコードがネイティブコードに変換された後、圧縮されて記憶される。このため、ＪＩＴを用いた従来のＪａｖａ（Ｒ）ＶＭに比べ、ＲＡＭの使用容量を削減することが可能になる。

【０２４０】

また、ネイティブコード格納領域３２に複数のメソッドのネイティブコードを格納することができるため、第７の実施の形態に比べて、処理が高速である。

【０２４１】

なお、実施の形態１～１４に記載の発明は、Ｊａｖａ（Ｒ）言語以外のオブジェクト指向言語で記載され、かつ中間言語にコンパイルされたプログラムにも適用可能である。

【０２４２】

今回開示された実施の形態はすべての点で例示であって制限的なものではない

と考えられるべきである。本発明の範囲は上記した説明ではなくて特許請求の範囲によって示され、特許請求の範囲と均等の意味および範囲内でのすべての変更が含まれることが意図される。

【０２４３】

【発明の効果】

オブジェクト指向言語のプログラムを実行する際に必要なメモリの容量を小さくすることができる。

【図面の簡単な説明】

【図１】　本発明の実施の形態に係る組込み機器のハードウェア構成を示すブロック図である。

【図２】　ＲＯＭに格納されているデータを説明するための図である。

【図３】　ＲＡＭに格納されているデータを説明するための図である。

【図４】　Ｊａｖａ（Ｒ）言語で記述されたプログラムの構成を示す図である。

【図５】　実施の形態１において、メソッドが呼出された場合の処理のフローチャートである。

【図６】　実施の形態１において、メソッドが呼出された場合の処理のフローチャートである。

【図７】　ＲＡＭに格納されているデータを説明するための図である。

【図８】　実施の形態２において、基本ブロックが呼出された場合の処理のフローチャートである。

【図９】　ＲＡＭに格納されるデータを説明するための図である。

【図１０】　実施の形態３において、命令が呼出された場合の処理のフローチャートである。

【図１１】　実施の形態４において、メソッドが呼出された場合の処理のフローチャートである。

【図１２】　実施の形態５において、メソッドが呼出された場合の処理のフローチャートである。

【図１３】　実施の形態５において、メソッドが呼出された場合の処理のフ

ローチャートである。

【図１４】　ＲＯＭに格納されているデータを説明するための図である。

【図１５】　実施の形態６において、メソッドが呼出された場合の処理のフローチャートである。

【図１６】　実施の形態６において、メソッドが呼出された場合の処理のフローチャートである。

【図１７】　実施の形態６において、メソッドが呼出された場合の処理を説明するための図である。

【図１８】　実施の形態７において、メソッドが呼出された場合の処理のフローチャートである。

【図１９】　実施の形態７において、メソッドが呼出された場合の処理のフローチャートである。

【図２０】　実施の形態７において、メソッドが呼出された場合の処理のフローチャートである。

【図２１】　実施の形態７において、メソッドが呼出された場合の処理を説明するための図である。

【図２２】　実施の形態８において、メソッドが呼出された場合の処理のフローチャートである。

【図２３】　実施の形態８において、メソッドが呼出された場合の処理のフローチャートである。

【図２４】　Ｊａｖａ（Ｒ）言語で記述されたプログラムの構成を示す図である。

【図２５】　実施の形態８において、メソッドが呼出された場合の処理のフローチャートである。

【図２６】　実施の形態８において、メソッドが呼出された場合の処理のフローチャートである。

【図２７】　実施の形態８において、メソッドが呼出された場合の処理を説明するための図である。

【図２８】　実施の形態９において、メソッドが呼出された場合の処理のフ

ローチャートである。

【図２９】　実施の形態９において、メソッドが呼出された場合の処理のフローチャートである。

【図３０】　実施の形態１０において、メソッドが呼出された場合の処理のフローチャートである。

【図３１】　実施の形態１０において、メソッドが呼出された場合の処理のフローチャートである。

【図３２】　実施の形態１１において、メソッドが呼出された場合の処理のフローチャートである。

【図３３】　実施の形態１１において、メソッドが呼出された場合の処理のフローチャートである。

【図３４】　実施の形態１２において、メソッドが呼出された場合の処理のフローチャートである。

【図３５】　実施の形態１２において、メソッドが呼出された場合の処理のフローチャートである。

【図３６】　実施の形態１３において、メソッドが呼出された場合の処理のフローチャートである。

【図３７】　実施の形態１３において、メソッドが呼出された場合の処理のフローチャートである。

【図３８】　実施の形態１４において、メソッドが呼出された場合の処理のフローチャートである。

【図３９】　実施の形態１４において、メソッドが呼出された場合の処理のフローチャートである。

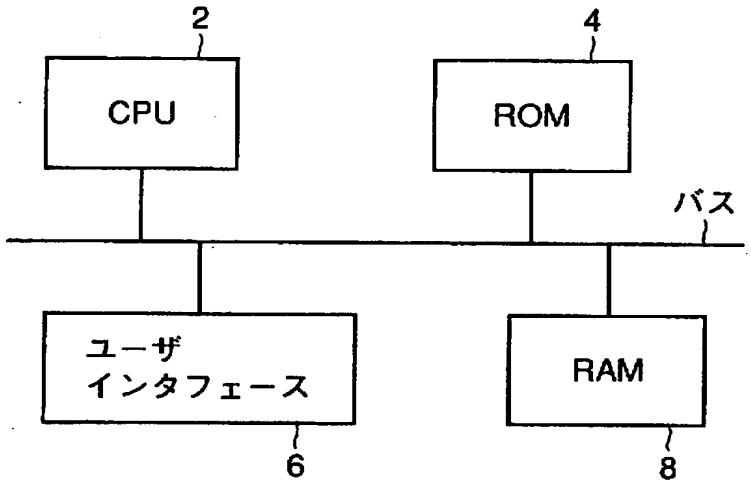
【符号の説明】

２　ＣＰＵ、４　ＲＯＭ、６　ユーザインタフェース、１２　圧縮バイトコード格納領域、１４，１２４　領域、１６　オペレーティングシステム格納領域、１８　モジュール格納領域、２０　インタプリタ格納領域、２２　ＪＩＴ格納領域、２４　圧縮モジュール格納領域、２６　伸張モジュール格納領域、２８　コード格納領域、３０　バイトコード格納領域、３２　ネイティブコード格納領域

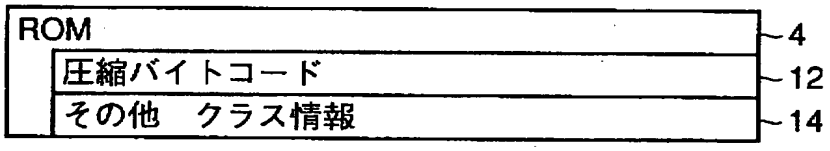
、３４　圧縮ネイティブコード格納領域、３６　メソッドステータス格納領域、３８，６２，９２　回数格納領域、４０，６４，９４　サイズ格納領域、４２，６６，９６　ネイティブコード有無フラグ格納領域、４４，６８，９８　コンパイル時刻（順番）格納領域、４６，７０，１００　圧縮情報格納領域、４８，７２，１０２　圧縮フラグ格納領域、５０，７４，１０４　伸張済みフラグ格納領域、５２，７６，１０６　圧縮方式格納領域、５４，７８，１０８　圧縮率格納領域、５６，８０，１１０　圧縮時刻（順番）格納領域、５８，８２，１１２　圧縮サイズ格納領域、６０　基本ブロックステータス格納領域、９０　命令ステータス格納領域、１２２　バイトコード格納領域。

【書類名】　　図面

【図１】

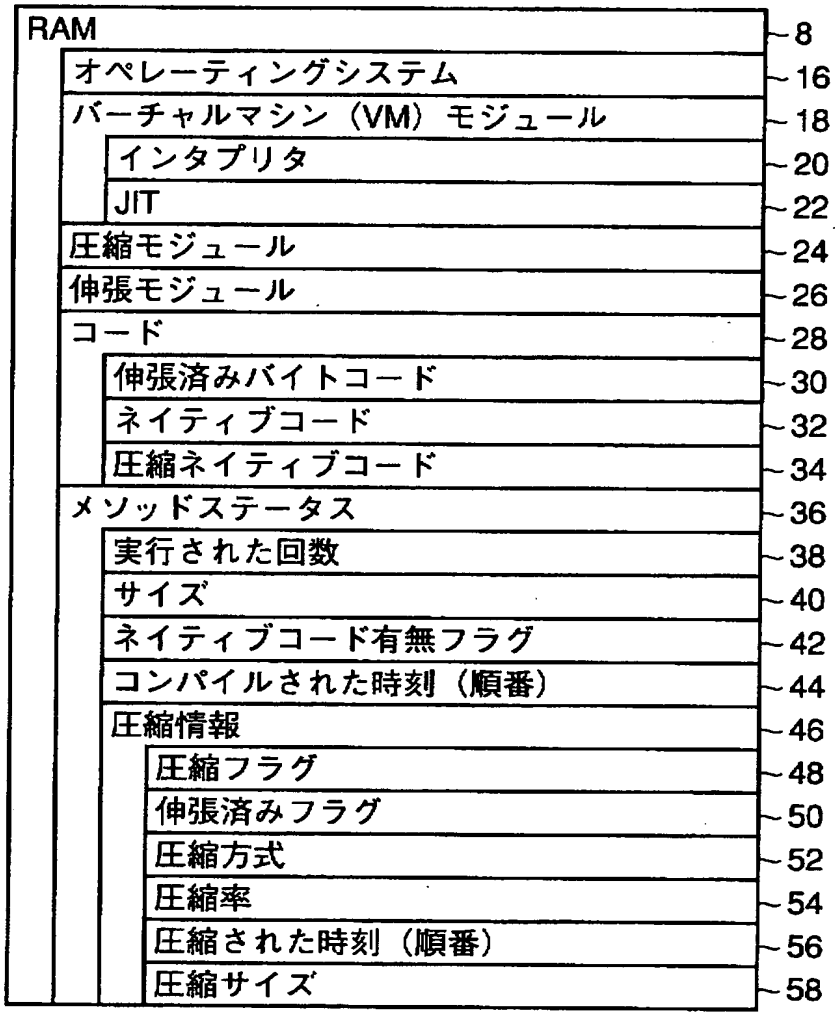


【図２】

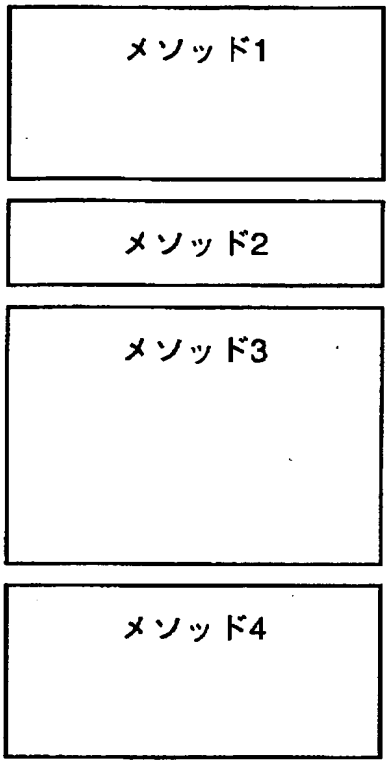


出証特２００１－３０４２６６８

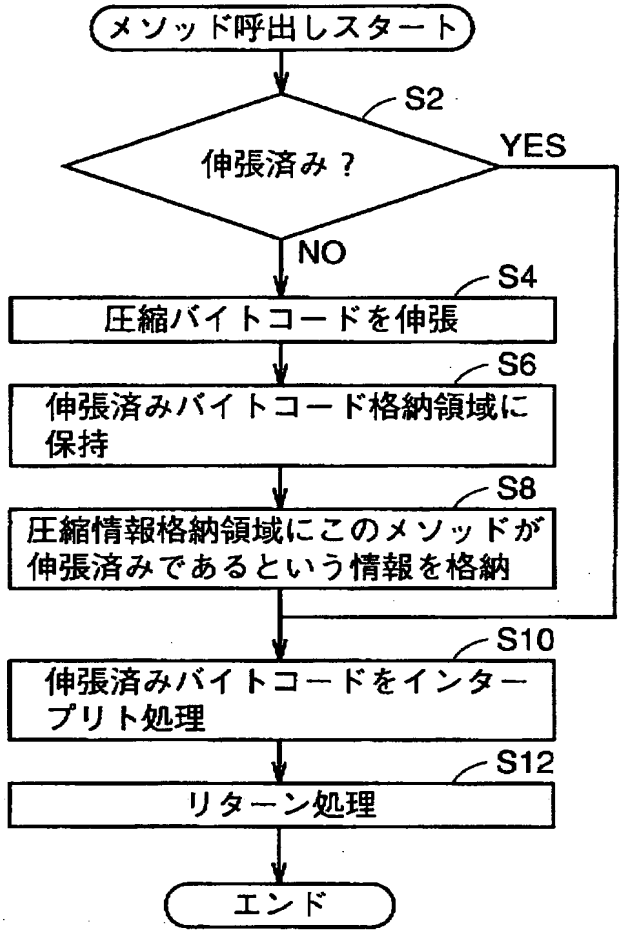
【図３】


RAM
8
オペレーティングシステム
16
バーチャルマシン（VM）モジュール
18
インタプリタ
20
JIT
22
圧縮モジュール
24
伸張モジュール
26
コード
28
伸張済みバイトコード
30
ネイティブコード
32
圧縮ネイティブコード
34
メソッドステータス
36
実行された回数
38
サイズ
40
ネイティブコード有無フラグ
42
コンパイルされた時刻（順番）
44
圧縮情報
46
圧縮フラグ
48
伸張済みフラグ
50
圧縮方式
52
圧縮率
54
圧縮された時刻（順番）
56
圧縮サイズ
58

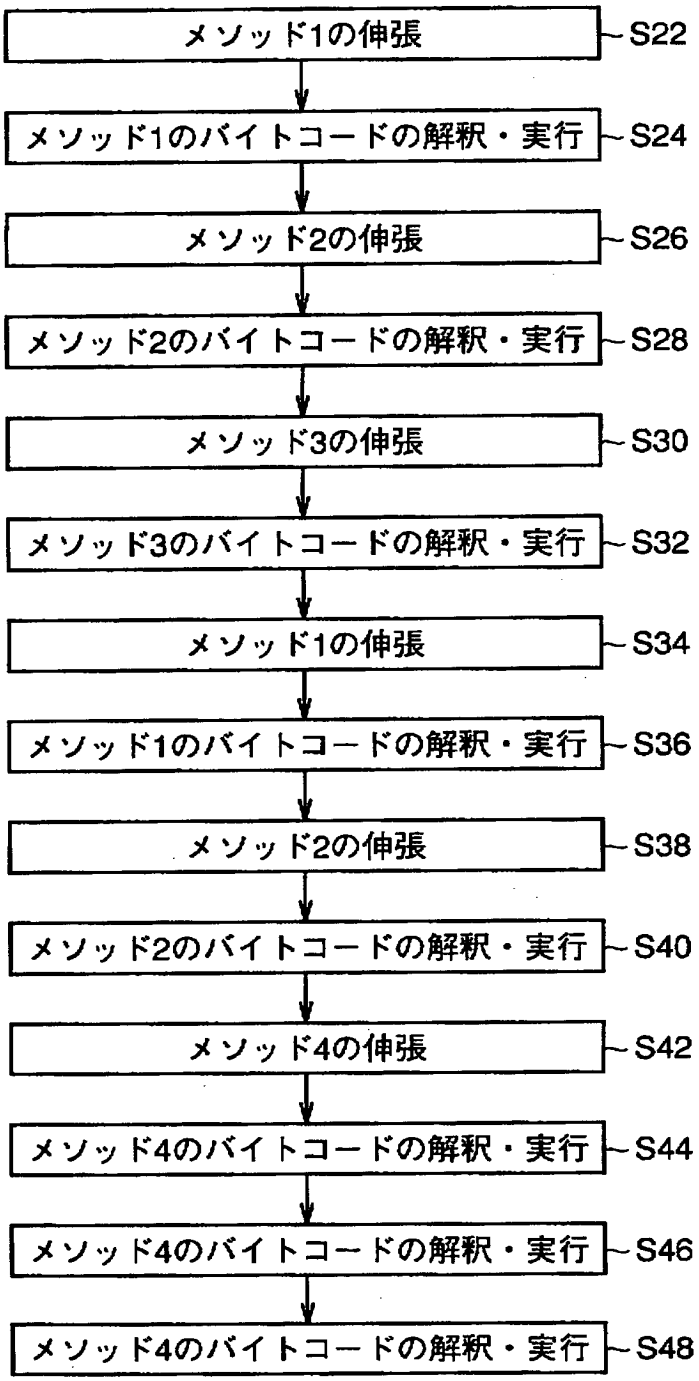
【図４】



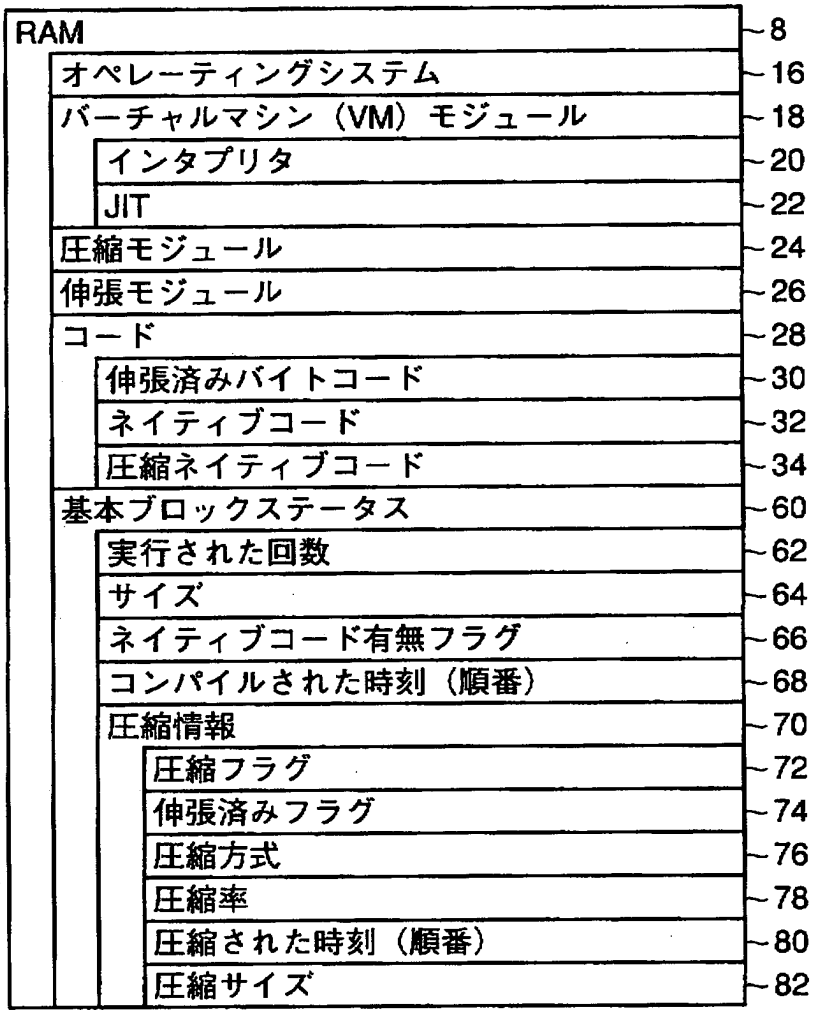
【図５】


メソッド呼出しスタート
S2
伸張済み？
YES
NO
S4
圧縮バイトコードを伸張
S6
伸張済みバイトコード格納領域に保持
S8
圧縮情報格納領域にこのメソッドが伸張済みであるという情報を格納
S10
伸張済みバイトコードをインタープリト処理
S12
リターン処理
エンド

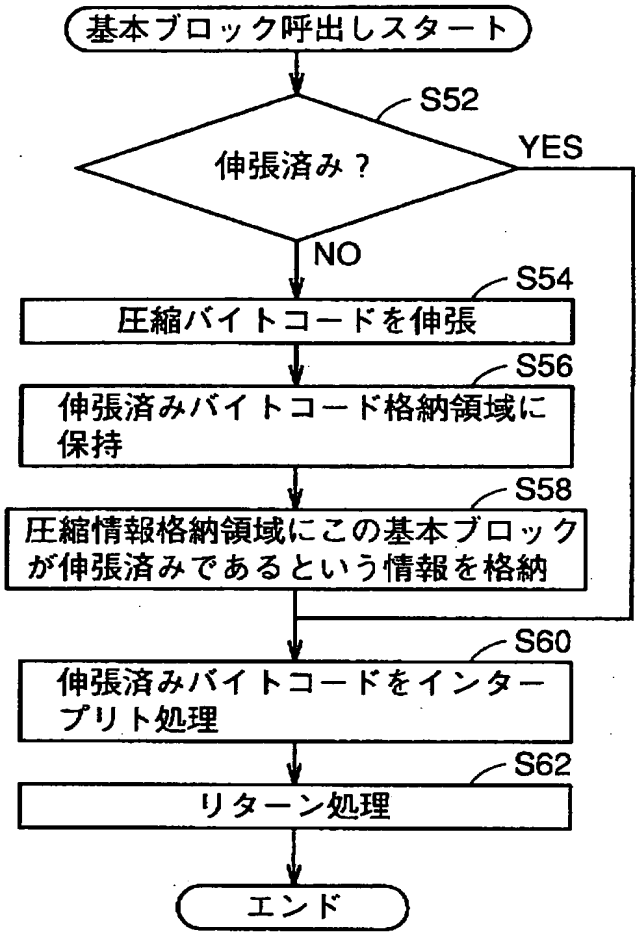
【図６】


メソッド1の伸張
S22
メソッド1のバイトコードの解釈・実行
S24
メソッド2の伸張
S26
メソッド2のバイトコードの解釈・実行
S28
メソッド3の伸張
S30
メソッド3のバイトコードの解釈・実行
S32
メソッド1の伸張
S34
メソッド1のバイトコードの解釈・実行
S36
メソッド2の伸張
S38
メソッド2のバイトコードの解釈・実行
S40
メソッド4の伸張
S42
メソッド4のバイトコードの解釈・実行
S44
メソッド4のバイトコードの解釈・実行
S46
メソッド4のバイトコードの解釈・実行
S48

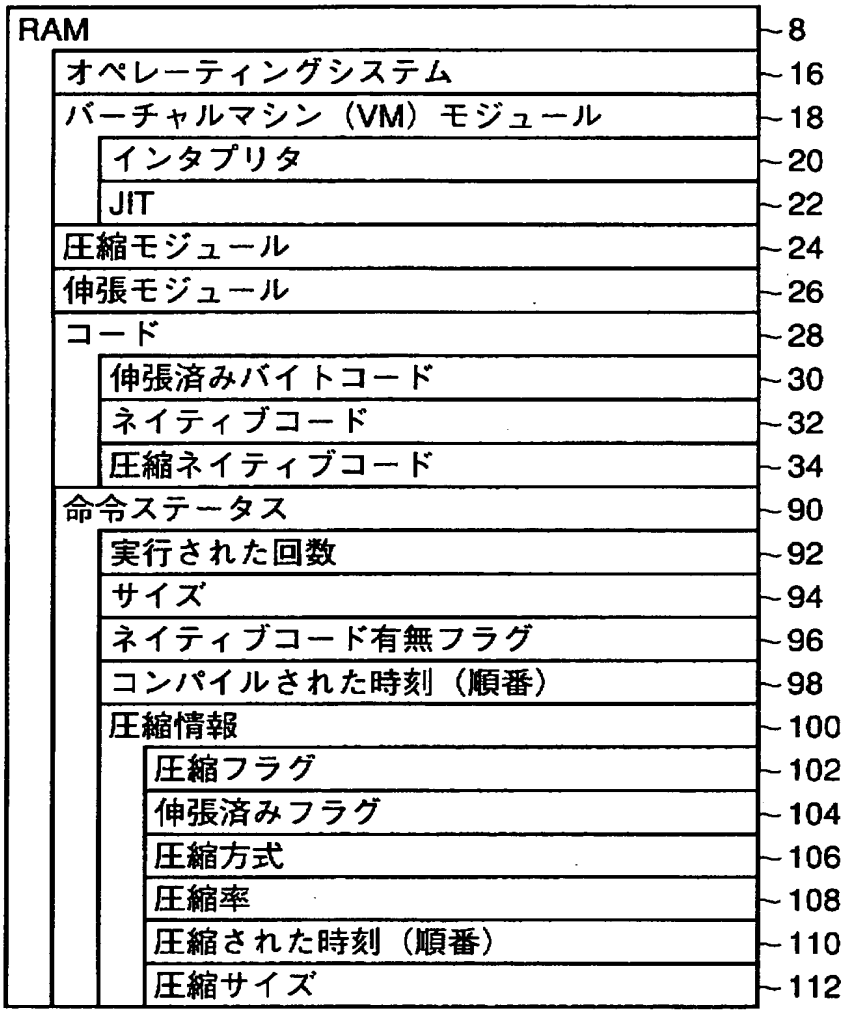
【図７】


RAM
8
オペレーティングシステム
16
バーチャルマシン（VM）モジュール
18
インタプリタ
20
JIT
22
圧縮モジュール
24
伸張モジュール
26
コード
28
伸張済みバイトコード
30
ネイティブコード
32
圧縮ネイティブコード
34
基本ブロックステータス
60
実行された回数
62
サイズ
64
ネイティブコード有無フラグ
66
コンパイルされた時刻（順番）
68
圧縮情報
70
圧縮フラグ
72
伸張済みフラグ
74
圧縮方式
76
圧縮率
78
圧縮された時刻（順番）
80
圧縮サイズ
82

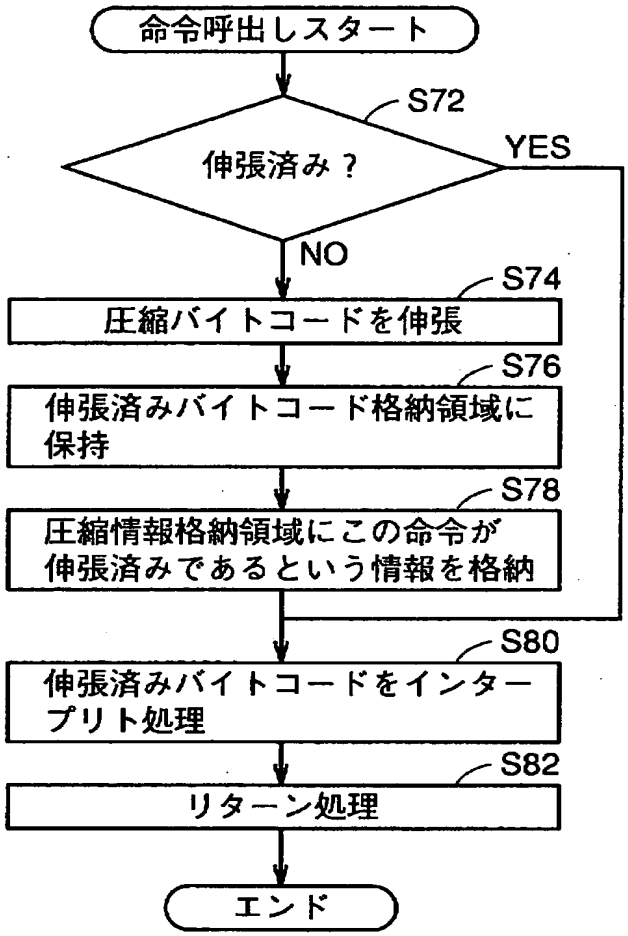
【図８】


基本ブロック呼出しスタート
S52
伸張済み？
YES
NO
S54
圧縮バイトコードを伸張
S56
伸張済みバイトコード格納領域に保持
S58
圧縮情報格納領域にこの基本ブロックが伸張済みであるという情報を格納
S60
伸張済みバイトコードをインタープリト処理
S62
リターン処理
エンド

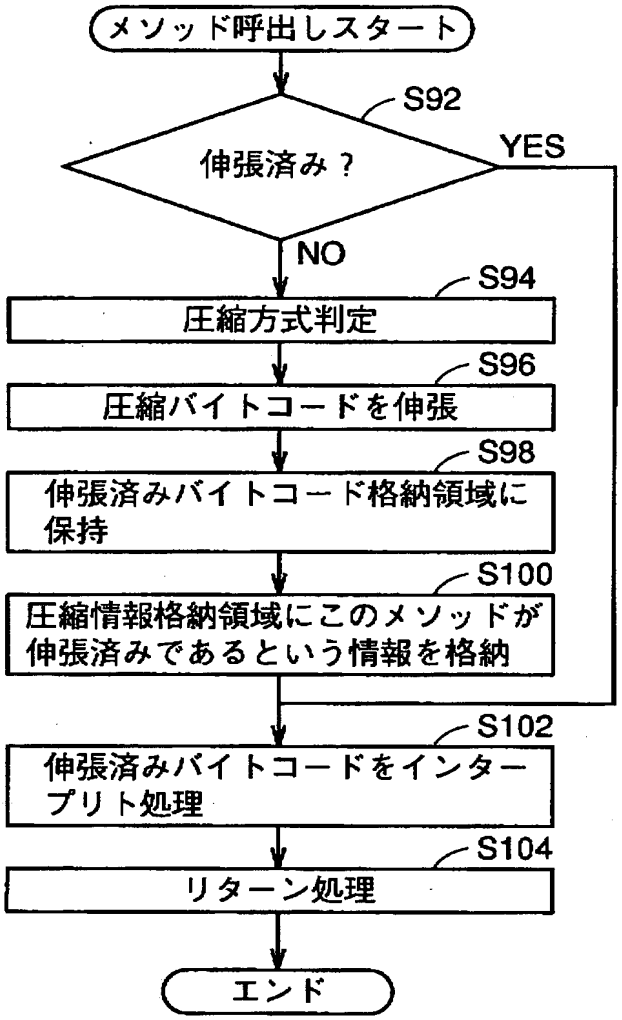
【図９】


RAM
8
オペレーティングシステム
16
バーチャルマシン（VM）モジュール
18
インタプリタ
20
JIT
22
圧縮モジュール
24
伸張モジュール
26
コード
28
伸張済みバイトコード
30
ネイティブコード
32
圧縮ネイティブコード
34
命令ステータス
90
実行された回数
92
サイズ
94
ネイティブコード有無フラグ
96
コンパイルされた時刻（順番）
98
圧縮情報
100
圧縮フラグ
102
伸張済みフラグ
104
圧縮方式
106
圧縮率
108
圧縮された時刻（順番）
110
圧縮サイズ
112

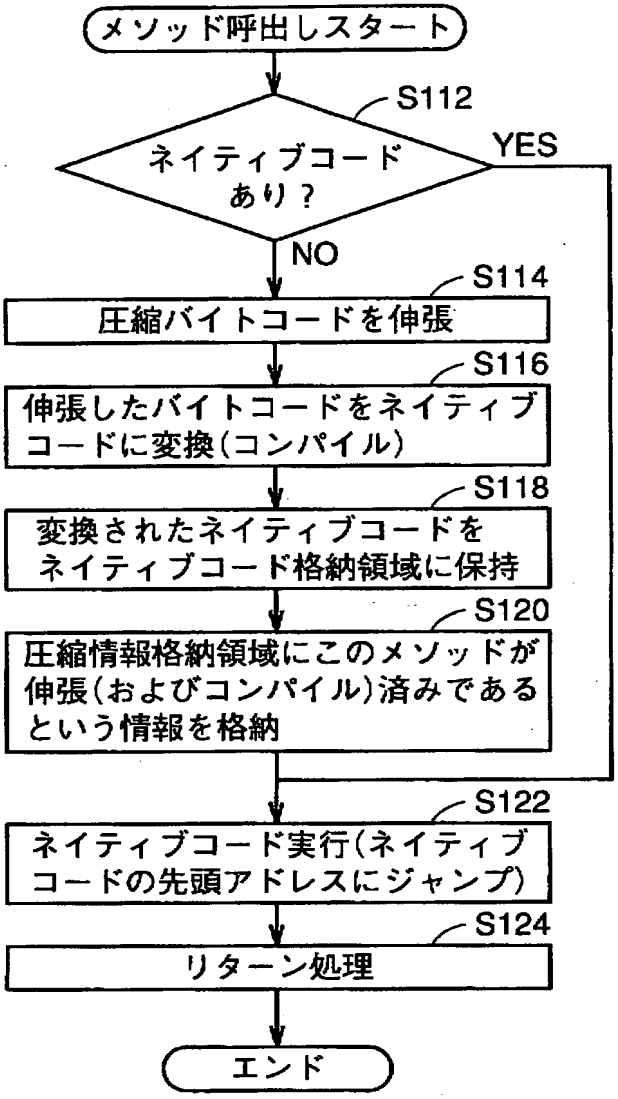
【図１０】


命令呼出しスタート
S72
伸張済み？
YES
NO
S74
圧縮バイトコードを伸張
S76
伸張済みバイトコード格納領域に保持
S78
圧縮情報格納領域にこの命令が伸張済みであるという情報を格納
S80
伸張済みバイトコードをインタープリト処理
S82
リターン処理
エンド

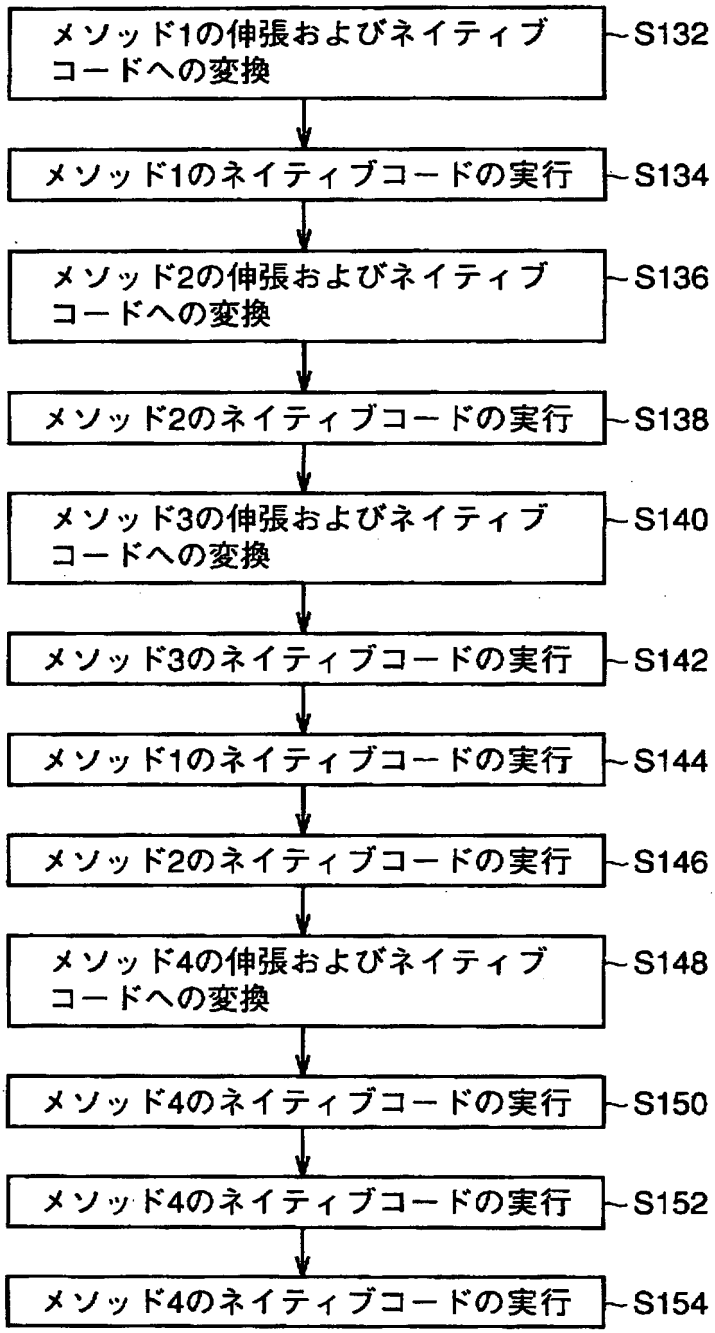
【図１１】


メソッド呼出しスタート
S92
伸張済み？
YES
NO
S94
圧縮方式判定
S96
圧縮バイトコードを伸張
S98
伸張済みバイトコード格納領域に保持
S100
圧縮情報格納領域にこのメソッドが伸張済みであるという情報を格納
S102
伸張済みバイトコードをインタープリト処理
S104
リターン処理
エンド

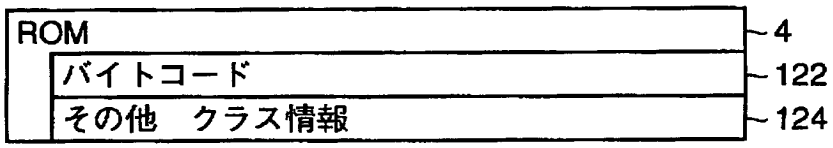
【図１２】


メソッド呼出しスタート
S112
ネイティブコード
あり？
YES
NO
S114
圧縮バイトコードを伸張
S116
伸張したバイトコードをネイティブ
コードに変換(コンパイル)
S118
変換されたネイティブコードを
ネイティブコード格納領域に保持
S120
圧縮情報格納領域にこのメソッドが
伸張(およびコンパイル)済みである
という情報を格納
S122
ネイティブコード実行(ネイティブ
コードの先頭アドレスにジャンプ)
S124
リターン処理
エンド

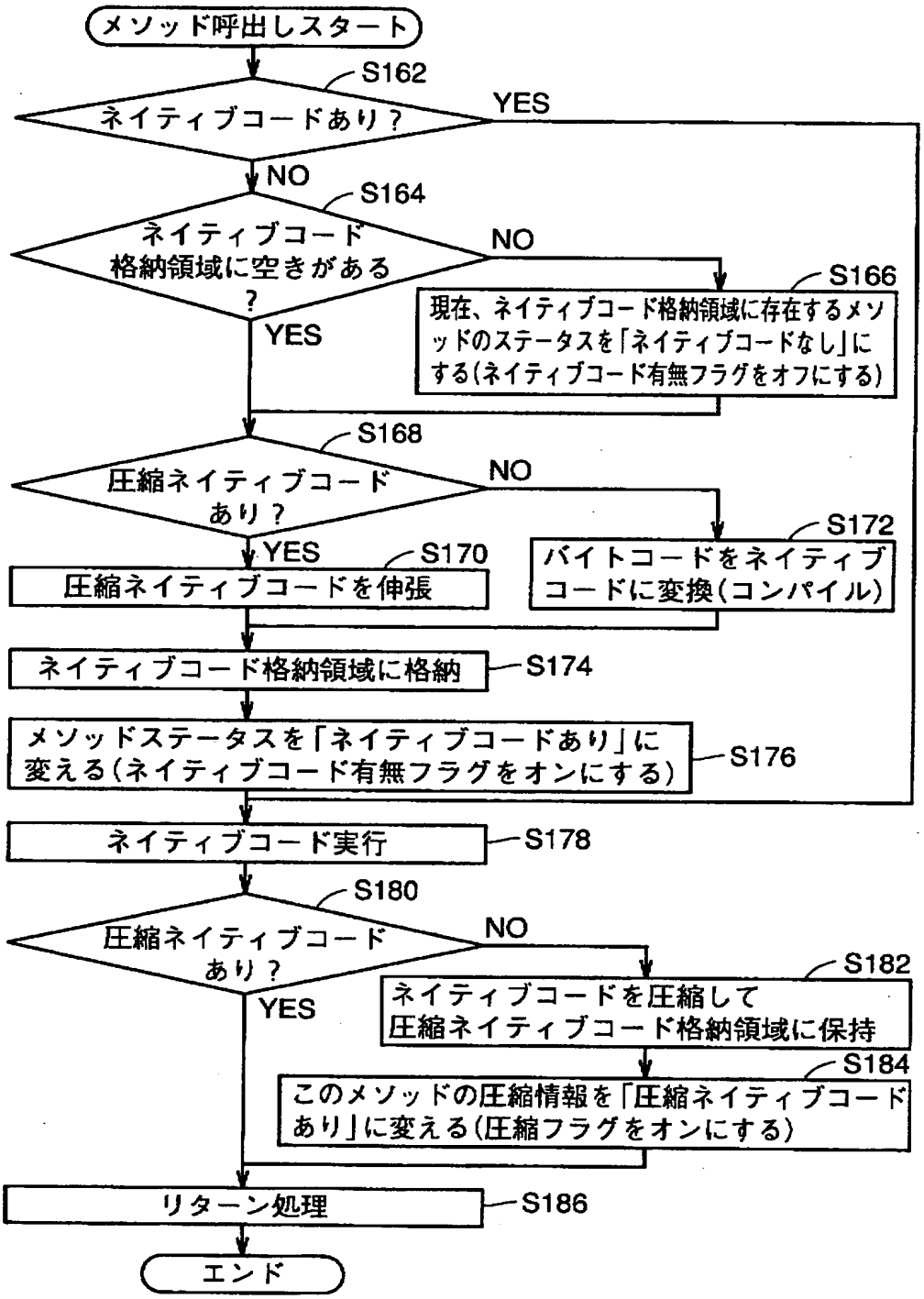
【図１３】


メソッド1の伸張およびネイティブコードへの変換
S132
メソッド1のネイティブコードの実行
S134
メソッド2の伸張およびネイティブコードへの変換
S136
メソッド2のネイティブコードの実行
S138
メソッド3の伸張およびネイティブコードへの変換
S140
メソッド3のネイティブコードの実行
S142
メソッド1のネイティブコードの実行
S144
メソッド2のネイティブコードの実行
S146
メソッド4の伸張およびネイティブコードへの変換
S148
メソッド4のネイティブコードの実行
S150
メソッド4のネイティブコードの実行
S152
メソッド4のネイティブコードの実行
S154

【図１４】



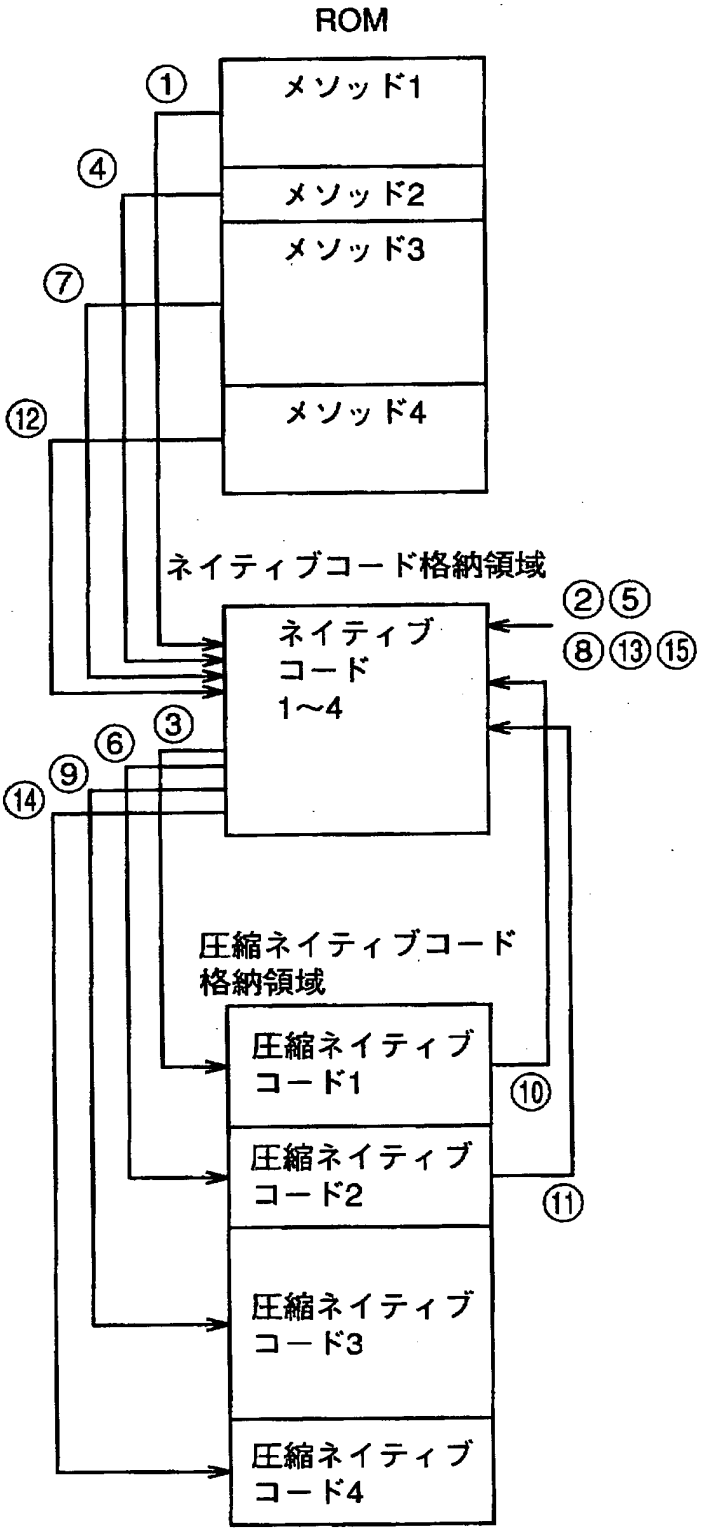
【図１５】


メソッド呼出しスタート
S162
ネイティブコードあり？
YES
NO
S164
ネイティブコード格納領域に空きがある？
NO
S166
現在、ネイティブコード格納領域に存在するメソッドのステータスを「ネイティブコードなし」にする(ネイティブコード有無フラグをオフにする)
YES
S168
圧縮ネイティブコードあり？
NO
S172
バイトコードをネイティブコードに変換(コンパイル)
YES
S170
圧縮ネイティブコードを伸張
ネイティブコード格納領域に格納
S174
メソッドステータスを「ネイティブコードあり」に変える(ネイティブコード有無フラグをオンにする)
S176
ネイティブコード実行
S178
S180
圧縮ネイティブコードあり？
NO
S182
ネイティブコードを圧縮して圧縮ネイティブコード格納領域に保持
YES
S184
このメソッドの圧縮情報を「圧縮ネイティブコードあり」に変える(圧縮フラグをオンにする)
リターン処理
S186
エンド

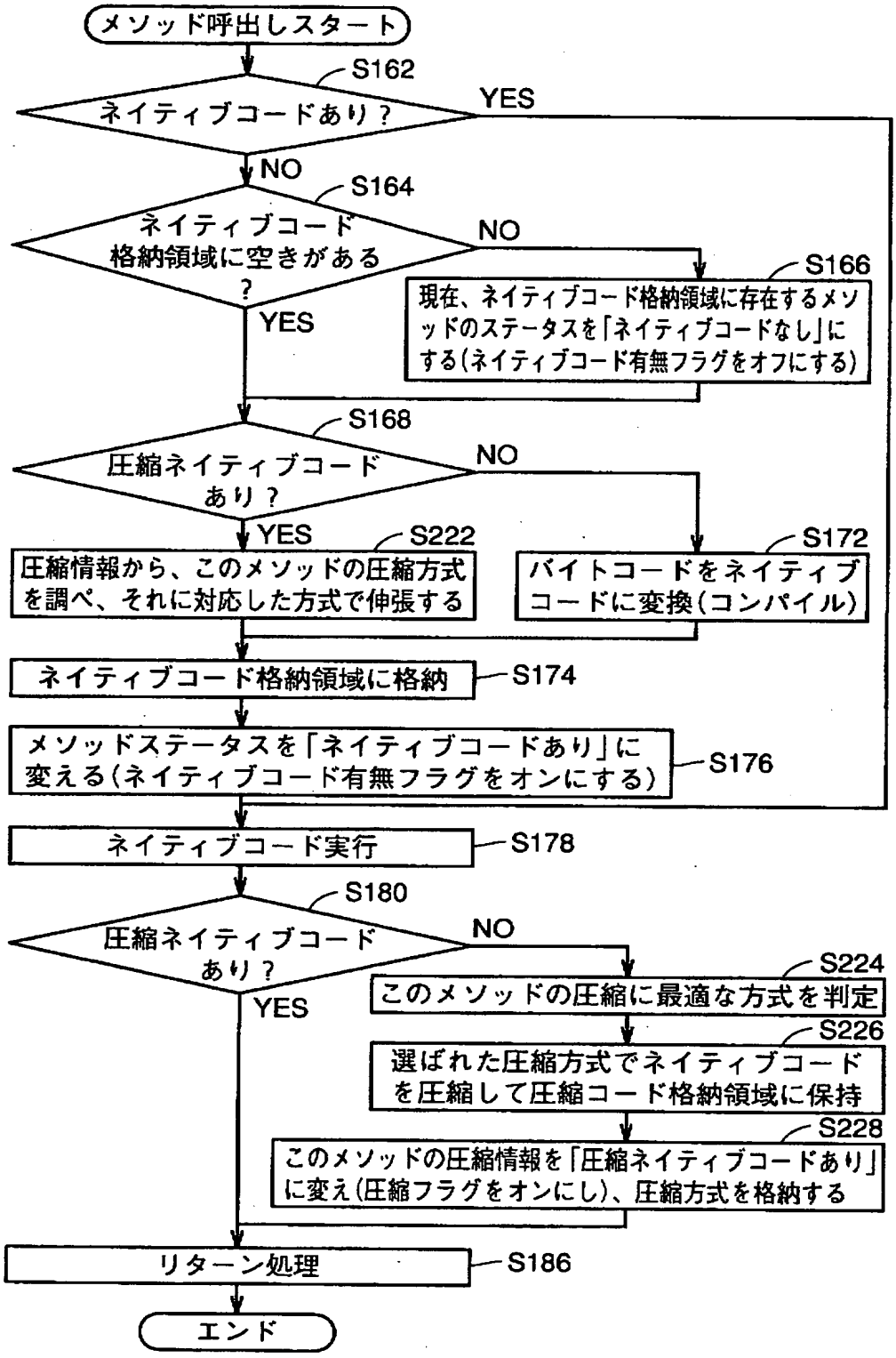
【図１６】


①メソッド1をネイティブコード1に変換
S192
②ネイティブコード1を実行
S194
③ネイティブコード1を圧縮
S196
④メソッド2をネイティブコード2に変換
S198
⑤ネイティブコード2を実行
S200
⑥ネイティブコード2を圧縮
S202
⑦メソッド3をネイティブコード3に変換
S204
⑧ネイティブコード3を実行
S206
⑨ネイティブコード3を圧縮
S208
⑩圧縮ネイティブコード1を伸張して実行
S210
⑪圧縮ネイティブコード2を伸張して実行
S212
⑫メソッド4をネイティブコード4に変換
S214
⑬ネイティブコード4を実行
S216
⑭ネイティブコード4を圧縮
S218
⑮ネイティブコード4を2回実行
S220

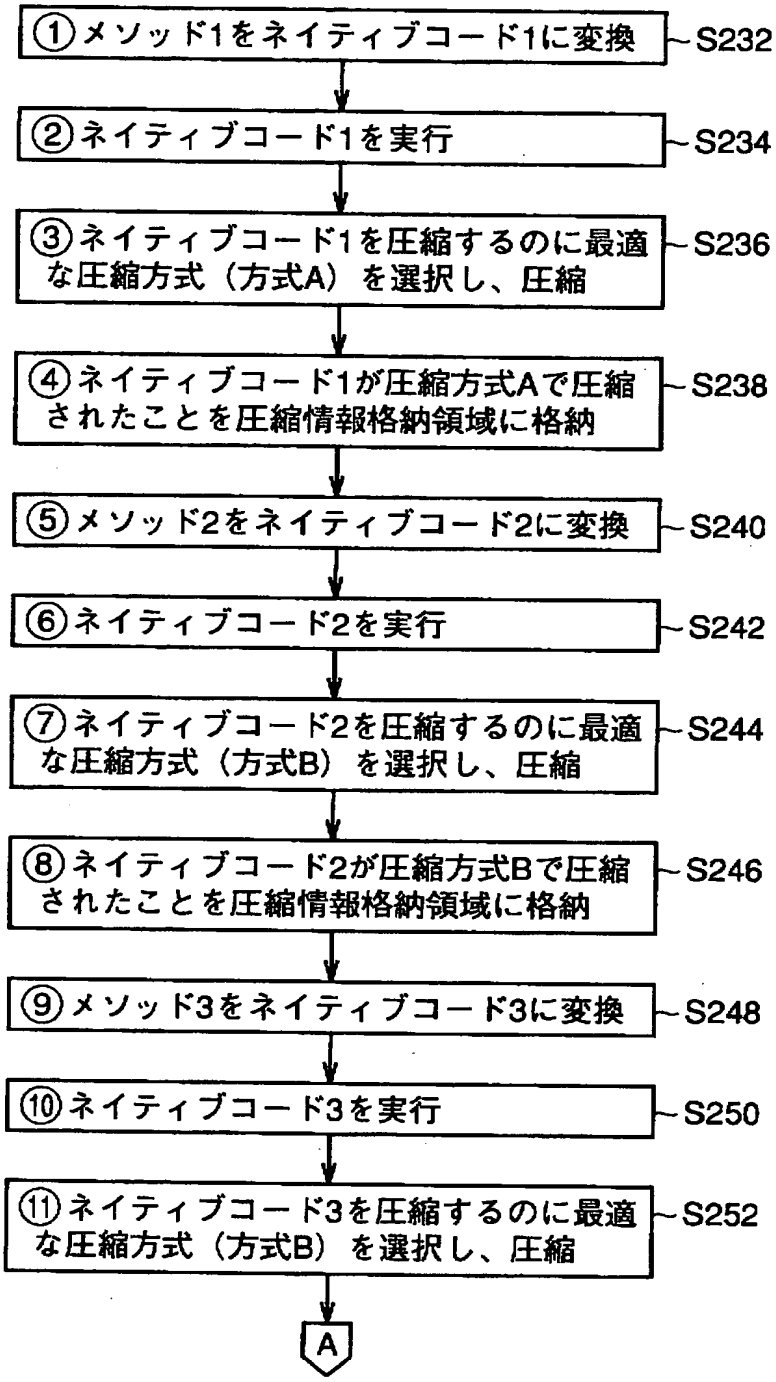
【図１７】


ROM
①
メソッド1
④
メソッド2
メソッド3
⑦
⑫
メソッド4
ネイティブコード格納領域
②⑤
⑧⑬⑮
ネイティブ
コード
1～4
③
⑥
⑨
⑭
圧縮ネイティブコード
格納領域
圧縮ネイティブ
コード1
⑩
圧縮ネイティブ
コード2
⑪
圧縮ネイティブ
コード3
圧縮ネイティブ
コード4

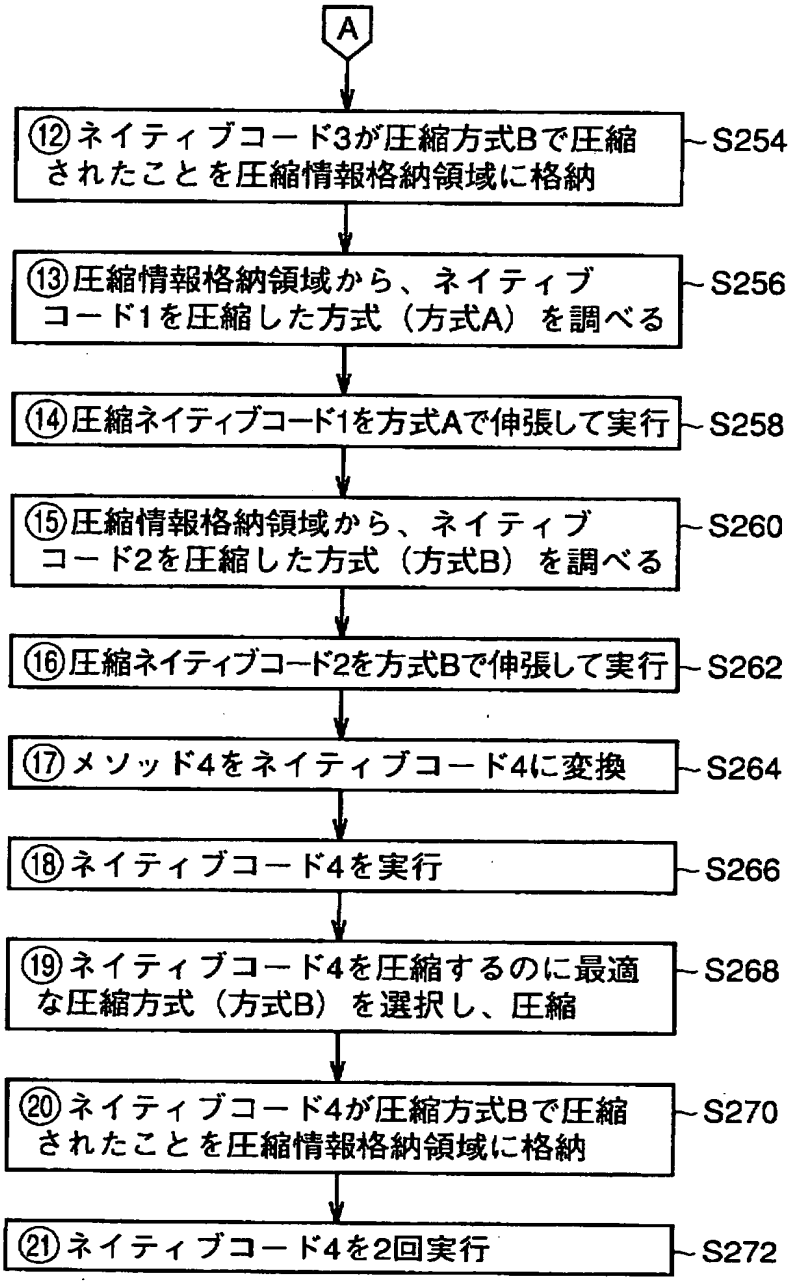
【図１８】


メソッド呼出しスタート
S162
ネイティブコードあり？
YES
NO
S164
ネイティブコード格納領域に空きがある？
NO
S166
現在、ネイティブコード格納領域に存在するメソッドのステータスを「ネイティブコードなし」にする(ネイティブコード有無フラグをオフにする)
YES
S168
圧縮ネイティブコードあり？
NO
YES
S222
圧縮情報から、このメソッドの圧縮方式を調べ、それに対応した方式で伸張する
S172
バイトコードをネイティブコードに変換(コンパイル)
ネイティブコード格納領域に格納
S174
メソッドステータスを「ネイティブコードあり」に変える(ネイティブコード有無フラグをオンにする)
S176
ネイティブコード実行
S178
S180
圧縮ネイティブコードあり？
NO
YES
S224
このメソッドの圧縮に最適な方式を判定
S226
選ばれた圧縮方式でネイティブコードを圧縮して圧縮コード格納領域に保持
S228
このメソッドの圧縮情報を「圧縮ネイティブコードあり」に変え(圧縮フラグをオンにし)、圧縮方式を格納する
リターン処理
S186
エンド

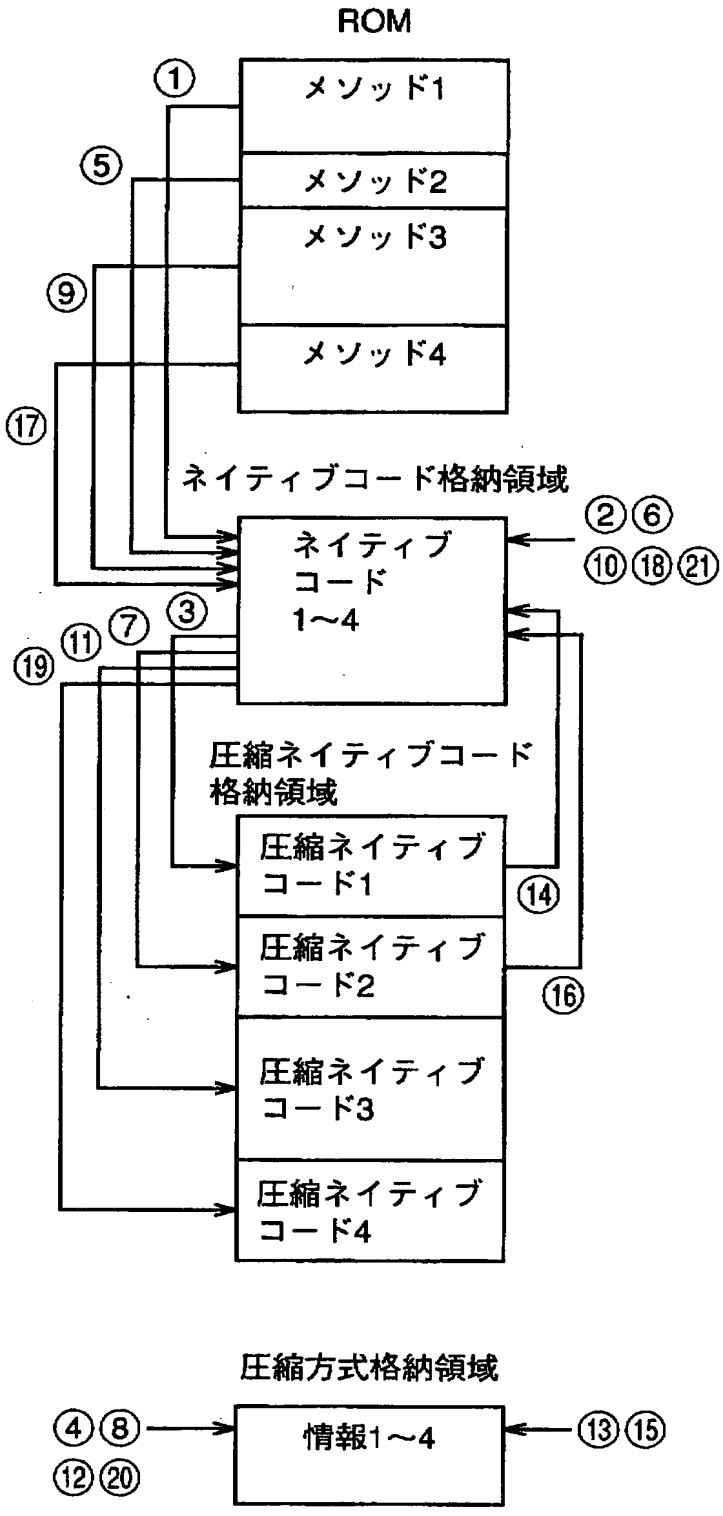
【図１９】


①メソッド1をネイティブコード1に変換
S232
②ネイティブコード1を実行
S234
③ネイティブコード1を圧縮するのに最適な圧縮方式（方式A）を選択し、圧縮
S236
④ネイティブコード1が圧縮方式Aで圧縮されたことを圧縮情報格納領域に格納
S238
⑤メソッド2をネイティブコード2に変換
S240
⑥ネイティブコード2を実行
S242
⑦ネイティブコード2を圧縮するのに最適な圧縮方式（方式B）を選択し、圧縮
S244
⑧ネイティブコード2が圧縮方式Bで圧縮されたことを圧縮情報格納領域に格納
S246
⑨メソッド3をネイティブコード3に変換
S248
⑩ネイティブコード3を実行
S250
⑪ネイティブコード3を圧縮するのに最適な圧縮方式（方式B）を選択し、圧縮
S252
A

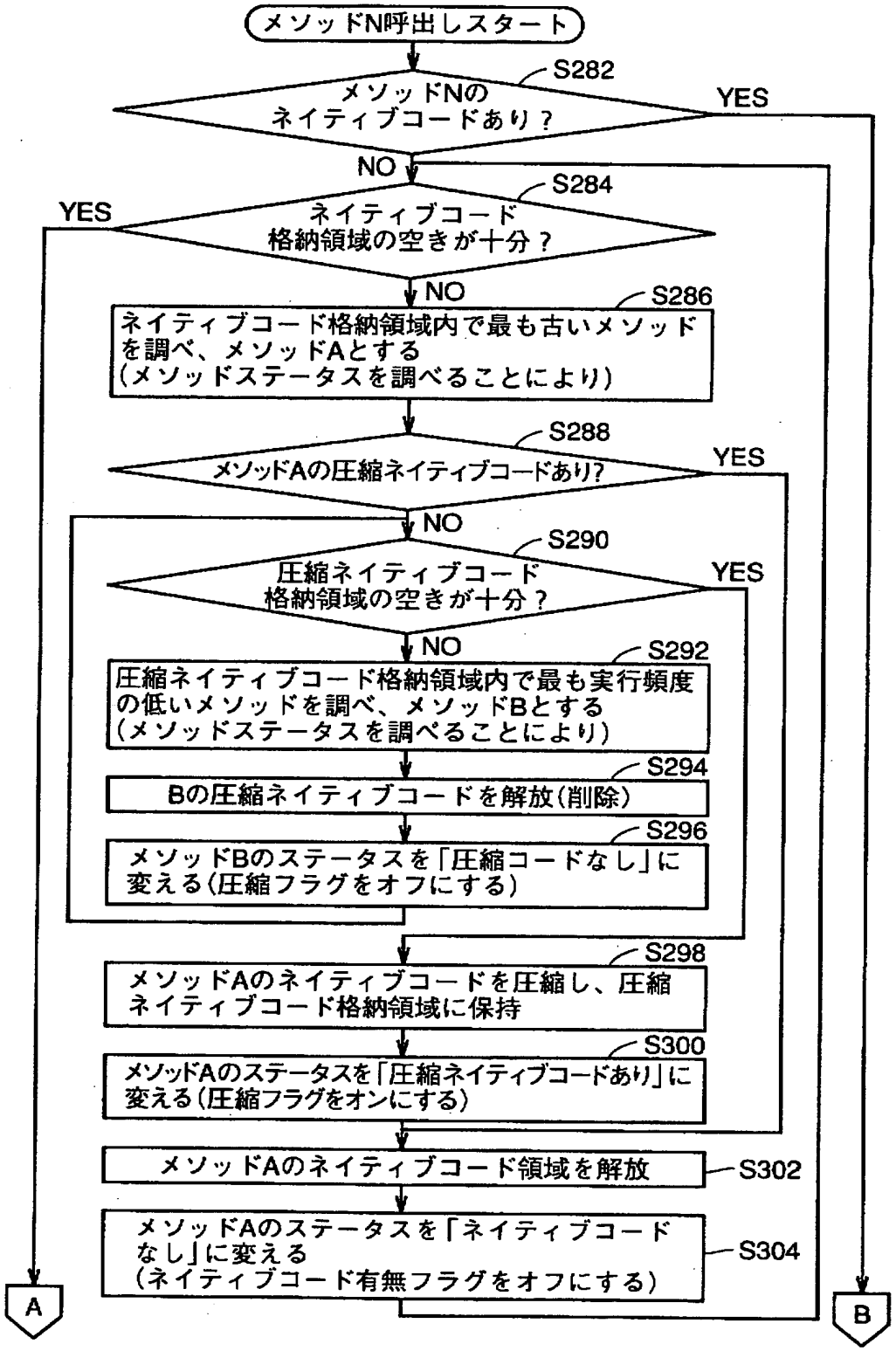
【図２０】


A
⑫ネイティブコード3が圧縮方式Bで圧縮されたことを圧縮情報格納領域に格納
S254
⑬圧縮情報格納領域から、ネイティブコード1を圧縮した方式（方式A）を調べる
S256
⑭圧縮ネイティブコード1を方式Aで伸張して実行
S258
⑮圧縮情報格納領域から、ネイティブコード2を圧縮した方式（方式B）を調べる
S260
⑯圧縮ネイティブコード2を方式Bで伸張して実行
S262
⑰メソッド4をネイティブコード4に変換
S264
⑱ネイティブコード4を実行
S266
⑲ネイティブコード4を圧縮するのに最適な圧縮方式（方式B）を選択し、圧縮
S268
⑳ネイティブコード4が圧縮方式Bで圧縮されたことを圧縮情報格納領域に格納
S270
㉑ネイティブコード4を2回実行
S272

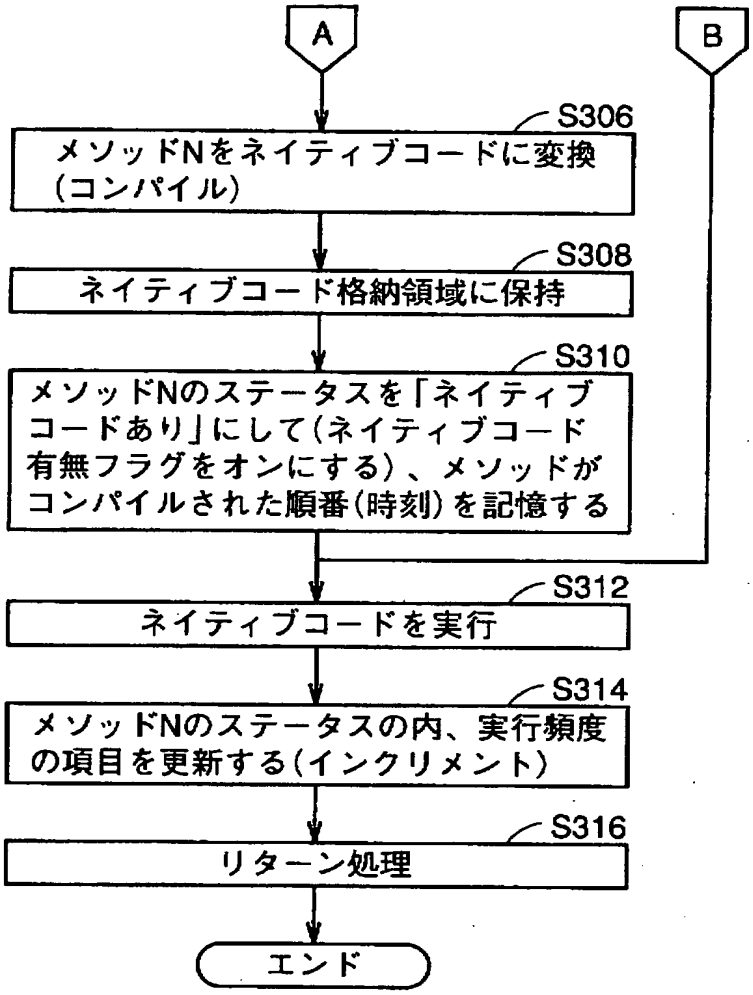
【図２１】


ROM
メソッド1
メソッド2
メソッド3
メソッド4
ネイティブコード格納領域
ネイティブ
コード
1～4
圧縮ネイティブコード
格納領域
圧縮ネイティブ
コード1
圧縮ネイティブ
コード2
圧縮ネイティブ
コード3
圧縮ネイティブ
コード4
圧縮方式格納領域
情報1～4
①
②⑥
③
④⑧
⑤
⑦
⑨
⑩⑱㉑
⑪
⑫⑳
⑬⑮
⑭
⑯
⑰
⑲

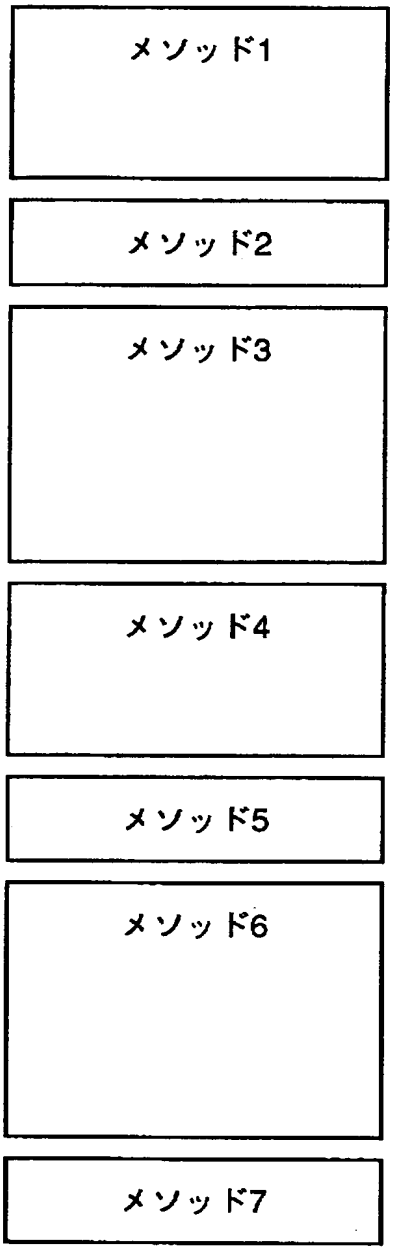
【図２２】


メソッドN呼出しスタート
S282
メソッドNの
ネイティブコードあり？
YES
NO
S284
ネイティブコード
格納領域の空きが十分？
YES
NO
S286
ネイティブコード格納領域内で最も古いメソッドを調べ、メソッドAとする
(メソッドステータスを調べることにより)
S288
メソッドAの圧縮ネイティブコードあり?
YES
NO
S290
圧縮ネイティブコード
格納領域の空きが十分？
YES
NO
S292
圧縮ネイティブコード格納領域内で最も実行頻度の低いメソッドを調べ、メソッドBとする
(メソッドステータスを調べることにより)
S294
Bの圧縮ネイティブコードを解放(削除)
S296
メソッドBのステータスを「圧縮コードなし」に変える(圧縮フラグをオフにする)
S298
メソッドAのネイティブコードを圧縮し、圧縮ネイティブコード格納領域に保持
S300
メソッドAのステータスを「圧縮ネイティブコードあり」に変える(圧縮フラグをオンにする)
メソッドAのネイティブコード領域を解放
S302
メソッドAのステータスを「ネイティブコードなし」に変える
(ネイティブコード有無フラグをオフにする)
S304
A
B

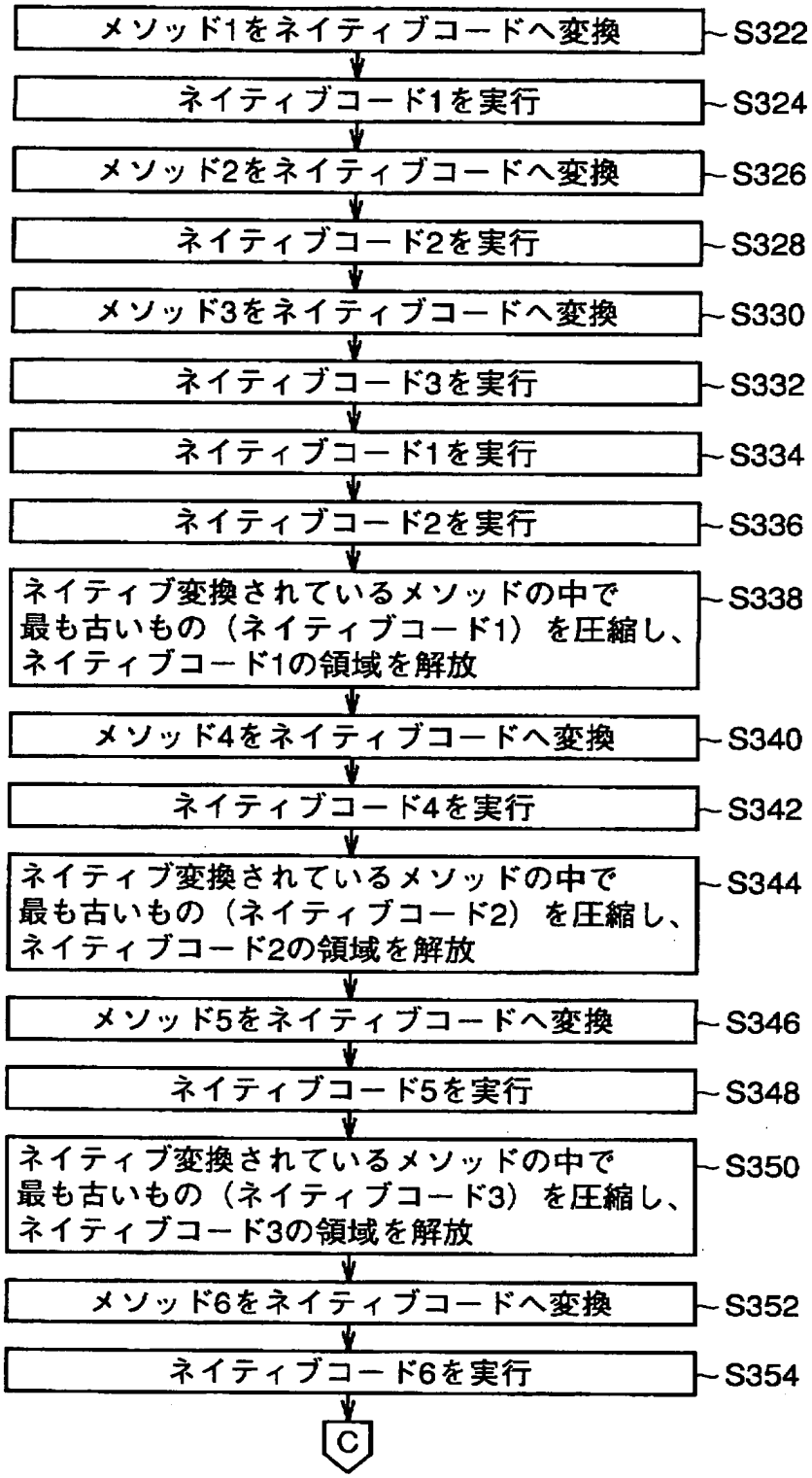
【図２３】


A
B
S306
メソッドNをネイティブコードに変換（コンパイル）
S308
ネイティブコード格納領域に保持
S310
メソッドNのステータスを「ネイティブコードあり」にして（ネイティブコード有無フラグをオンにする）、メソッドがコンパイルされた順番（時刻）を記憶する
S312
ネイティブコードを実行
S314
メソッドNのステータスの内、実行頻度の項目を更新する（インクリメント）
S316
リターン処理
エンド

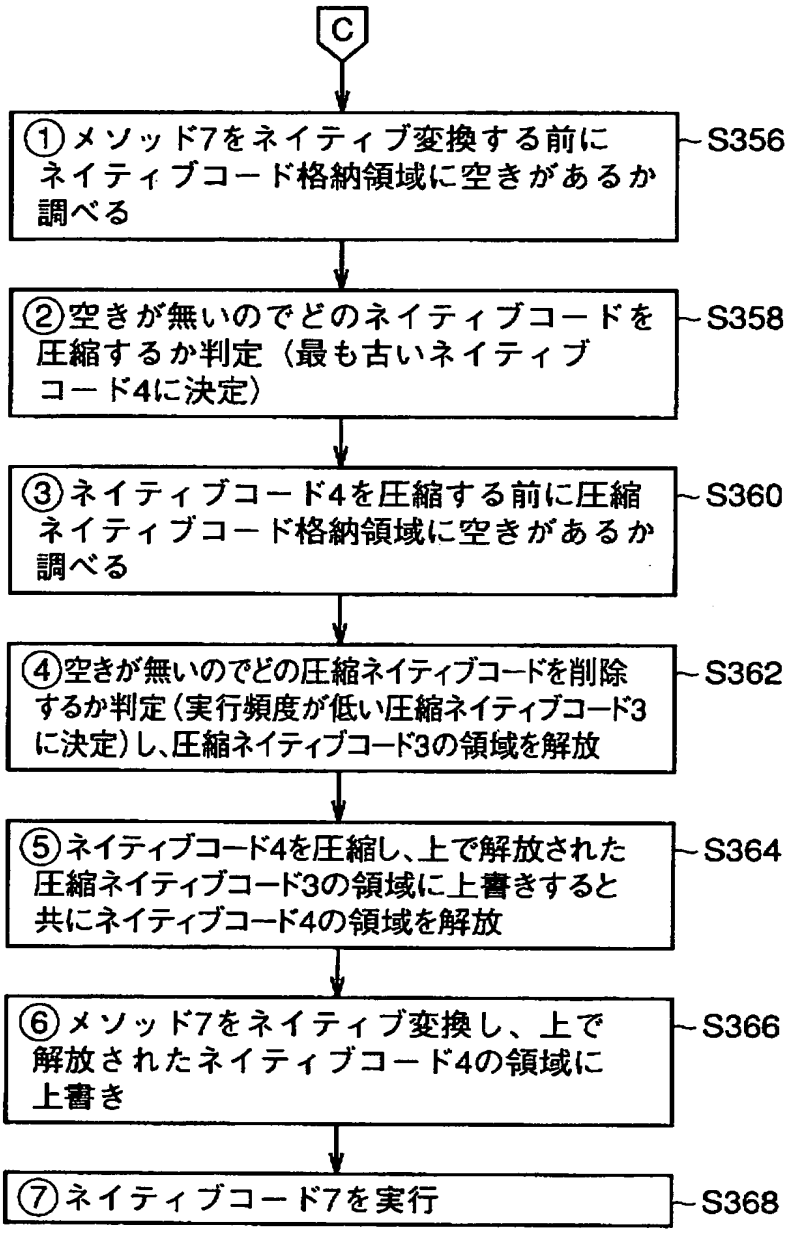
【図２４】



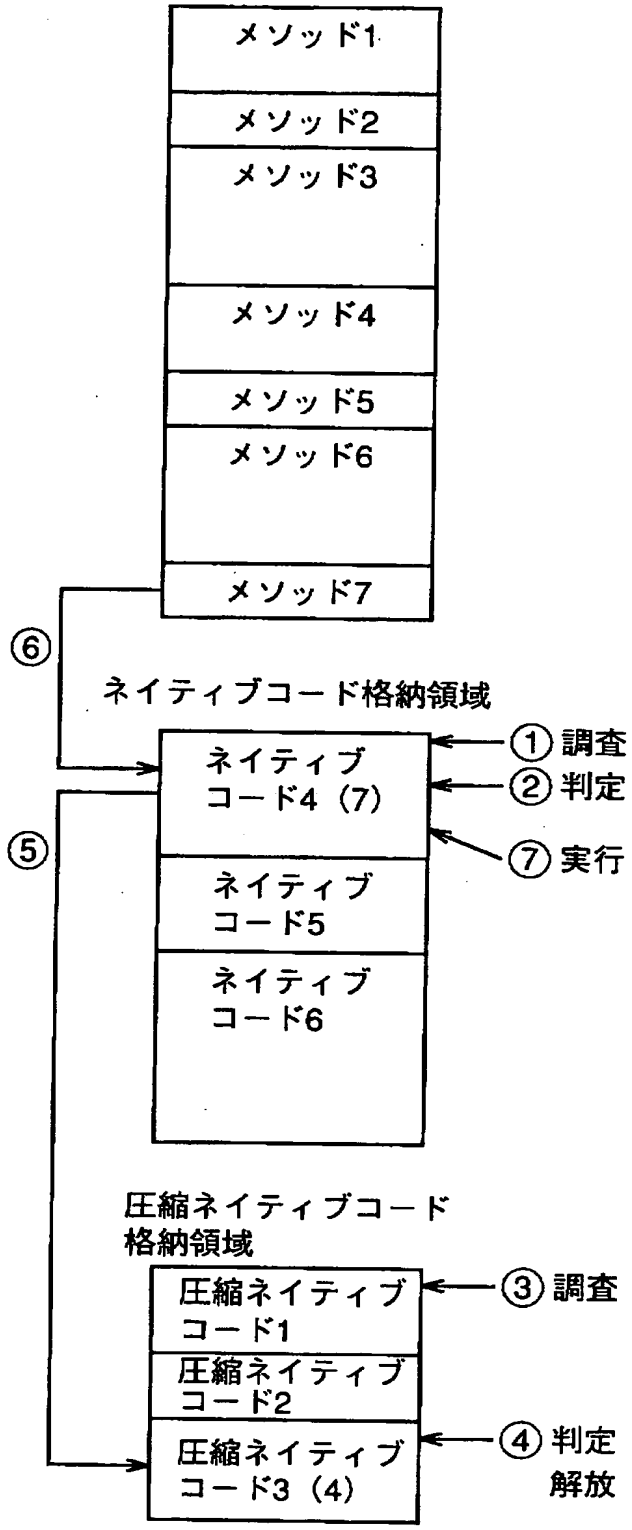
【図２５】


メソッド1をネイティブコードへ変換
S322
ネイティブコード1を実行
S324
メソッド2をネイティブコードへ変換
S326
ネイティブコード2を実行
S328
メソッド3をネイティブコードへ変換
S330
ネイティブコード3を実行
S332
ネイティブコード1を実行
S334
ネイティブコード2を実行
S336
ネイティブ変換されているメソッドの中で
最も古いもの（ネイティブコード1）を圧縮し、
ネイティブコード1の領域を解放
S338
メソッド4をネイティブコードへ変換
S340
ネイティブコード4を実行
S342
ネイティブ変換されているメソッドの中で
最も古いもの（ネイティブコード2）を圧縮し、
ネイティブコード2の領域を解放
S344
メソッド5をネイティブコードへ変換
S346
ネイティブコード5を実行
S348
ネイティブ変換されているメソッドの中で
最も古いもの（ネイティブコード3）を圧縮し、
ネイティブコード3の領域を解放
S350
メソッド6をネイティブコードへ変換
S352
ネイティブコード6を実行
S354
C

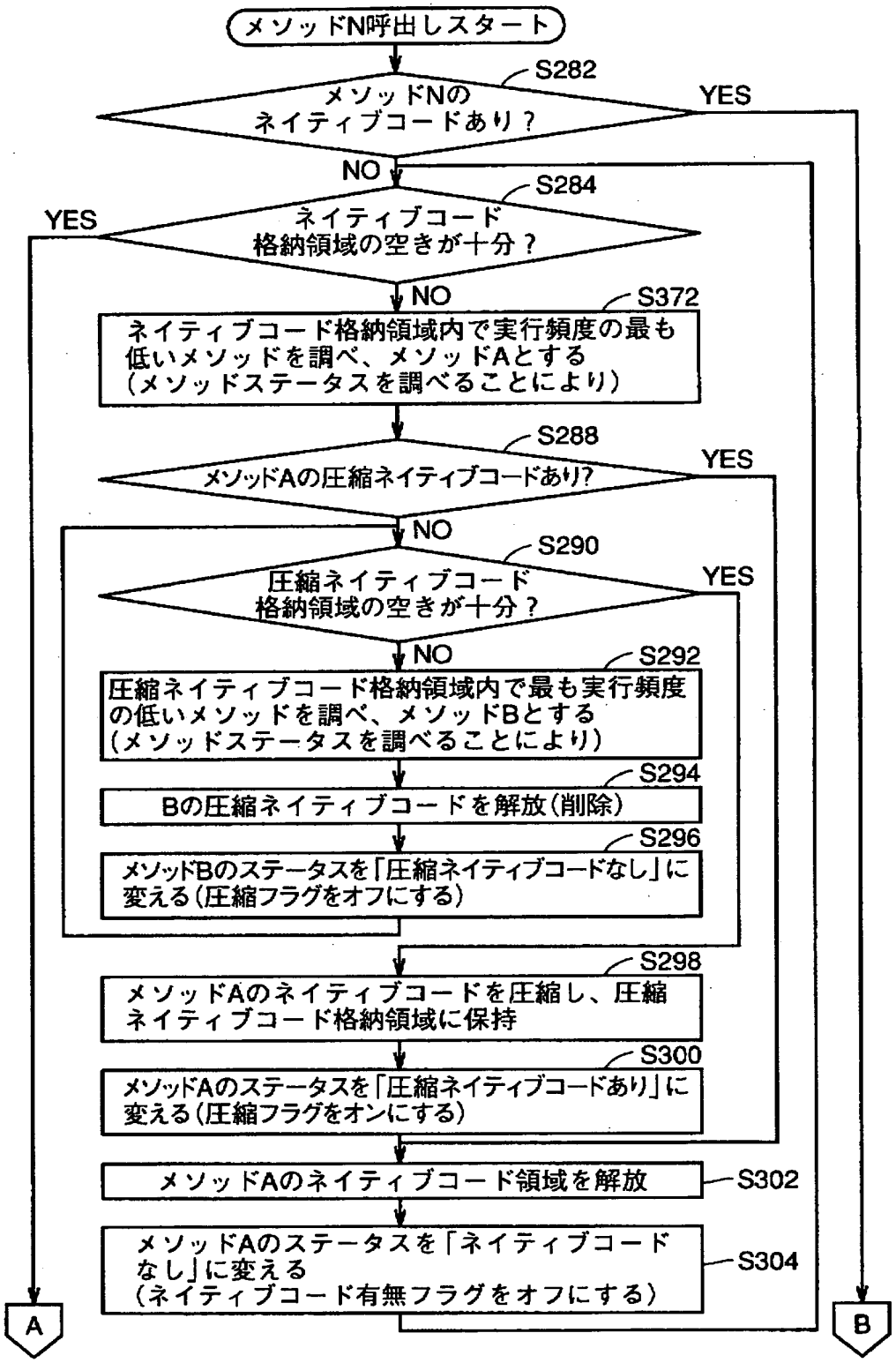
【図２６】


C
①メソッド7をネイティブ変換する前にネイティブコード格納領域に空きがあるか調べる
S356
②空きが無いのでどのネイティブコードを圧縮するか判定（最も古いネイティブコード4に決定）
S358
③ネイティブコード4を圧縮する前に圧縮ネイティブコード格納領域に空きがあるか調べる
S360
④空きが無いのでどの圧縮ネイティブコードを削除するか判定（実行頻度が低い圧縮ネイティブコード3に決定）し、圧縮ネイティブコード3の領域を解放
S362
⑤ネイティブコード4を圧縮し、上で解放された圧縮ネイティブコード3の領域に上書きすると共にネイティブコード4の領域を解放
S364
⑥メソッド7をネイティブ変換し、上で解放されたネイティブコード4の領域に上書き
S366
⑦ネイティブコード7を実行
S368

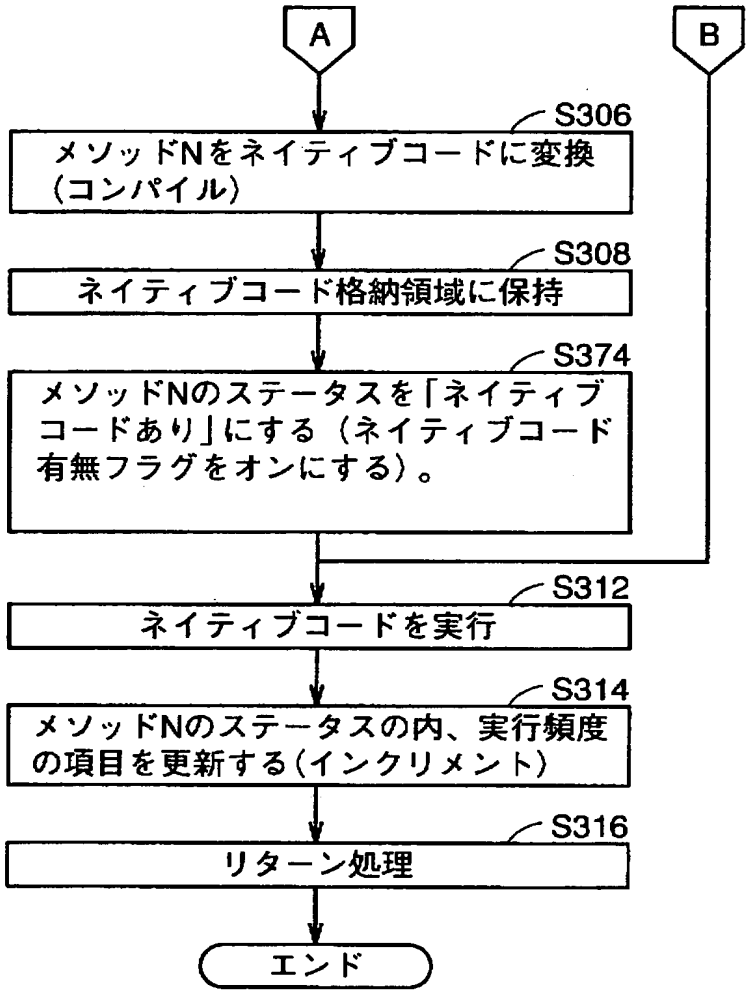
【図２７】


メソッド1
メソッド2
メソッド3
メソッド4
メソッド5
メソッド6
メソッド7
⑥
ネイティブコード格納領域
ネイティブ
コード4（7）
ネイティブ
コード5
ネイティブ
コード6
①調査
②判定
⑦実行
⑤
圧縮ネイティブコード
格納領域
圧縮ネイティブ
コード1
圧縮ネイティブ
コード2
圧縮ネイティブ
コード3（4）
③調査
④判定
解放

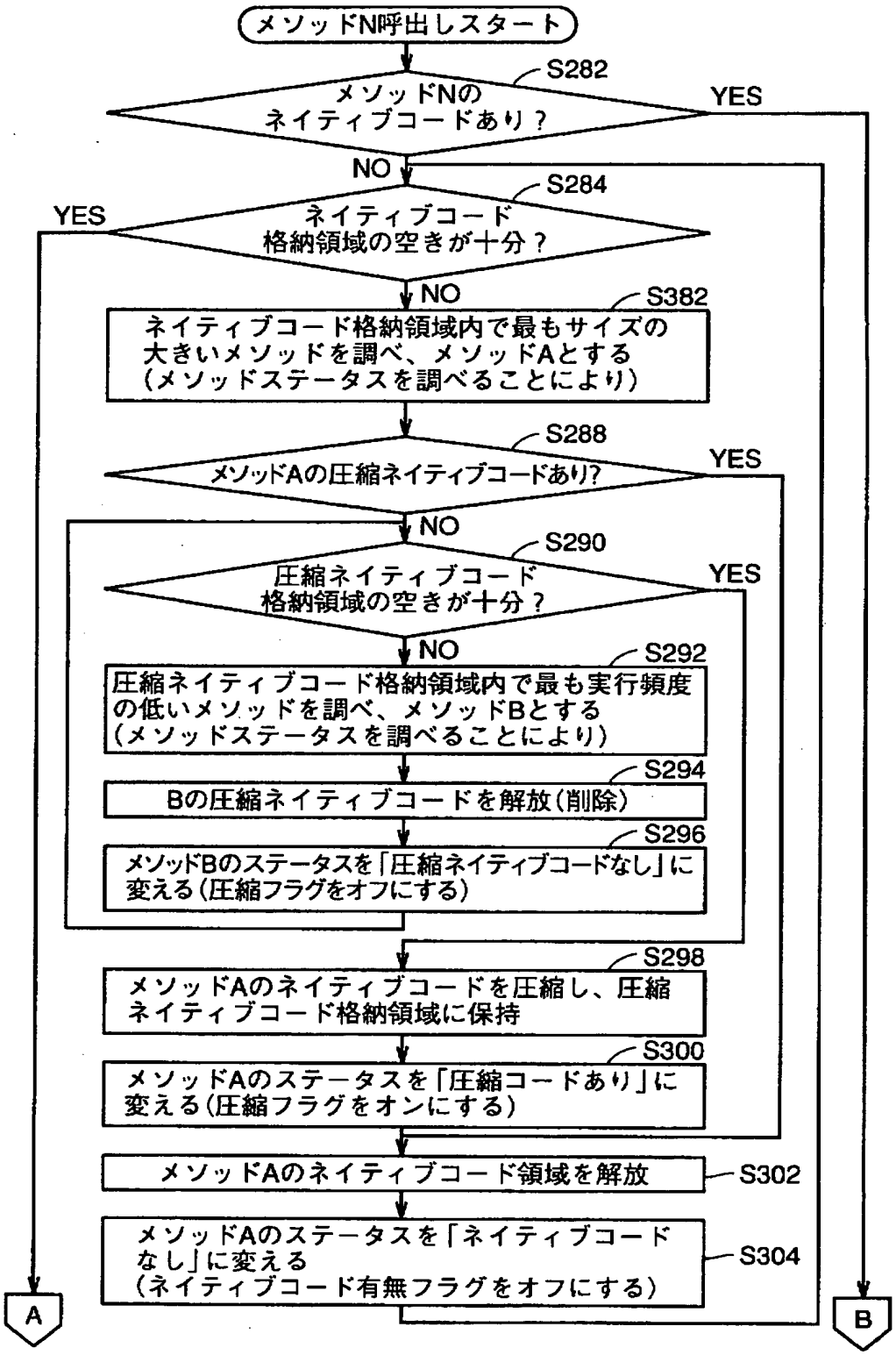
【図２８】


メソッドN呼出しスタート
S282
メソッドNの
ネイティブコードあり？
YES
NO
S284
ネイティブコード
格納領域の空きが十分？
YES
NO
S372
ネイティブコード格納領域内で実行頻度の最も
低いメソッドを調べ、メソッドAとする
（メソッドステータスを調べることにより）
S288
メソッドAの圧縮ネイティブコードあり?
YES
NO
S290
圧縮ネイティブコード
格納領域の空きが十分？
YES
NO
S292
圧縮ネイティブコード格納領域内で最も実行頻度
の低いメソッドを調べ、メソッドBとする
（メソッドステータスを調べることにより）
S294
Bの圧縮ネイティブコードを解放（削除）
S296
メソッドBのステータスを「圧縮ネイティブコードなし」に
変える（圧縮フラグをオフにする）
S298
メソッドAのネイティブコードを圧縮し、圧縮
ネイティブコード格納領域に保持
S300
メソッドAのステータスを「圧縮ネイティブコードあり」に
変える（圧縮フラグをオンにする）
メソッドAのネイティブコード領域を解放
S302
メソッドAのステータスを「ネイティブコード
なし」に変える
（ネイティブコード有無フラグをオフにする）
S304
A
B

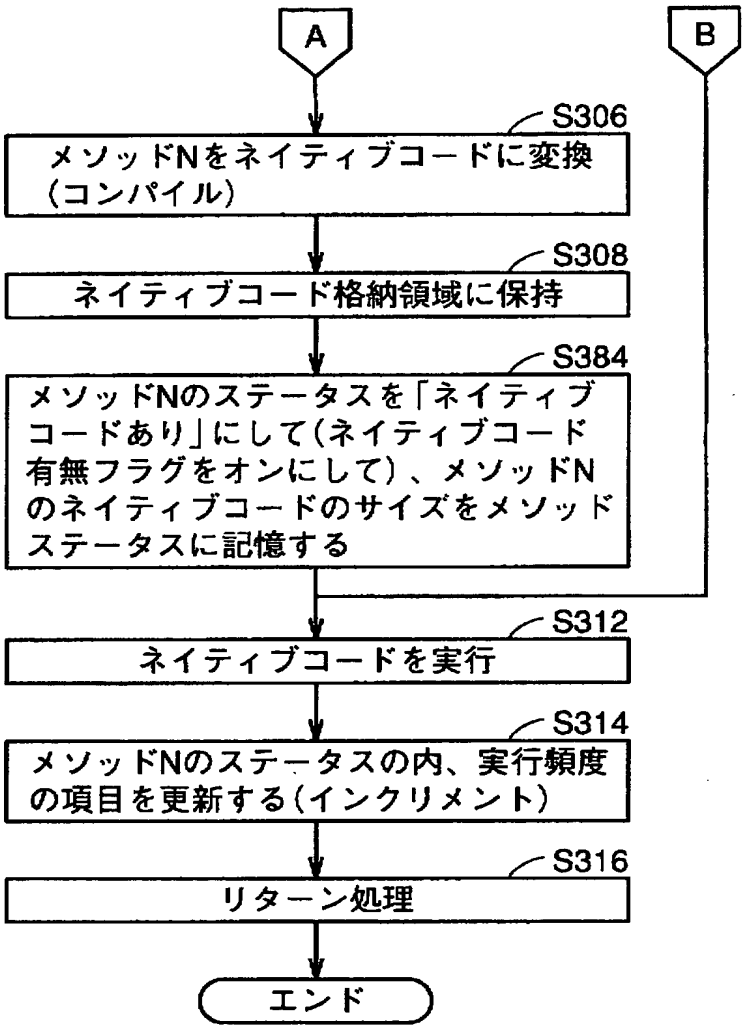
【図２９】


A
B
S306
メソッドNをネイティブコードに変換（コンパイル）
S308
ネイティブコード格納領域に保持
S374
メソッドNのステータスを「ネイティブコードあり」にする（ネイティブコード有無フラグをオンにする）。
S312
ネイティブコードを実行
S314
メソッドNのステータスの内、実行頻度の項目を更新する（インクリメント）
S316
リターン処理
エンド

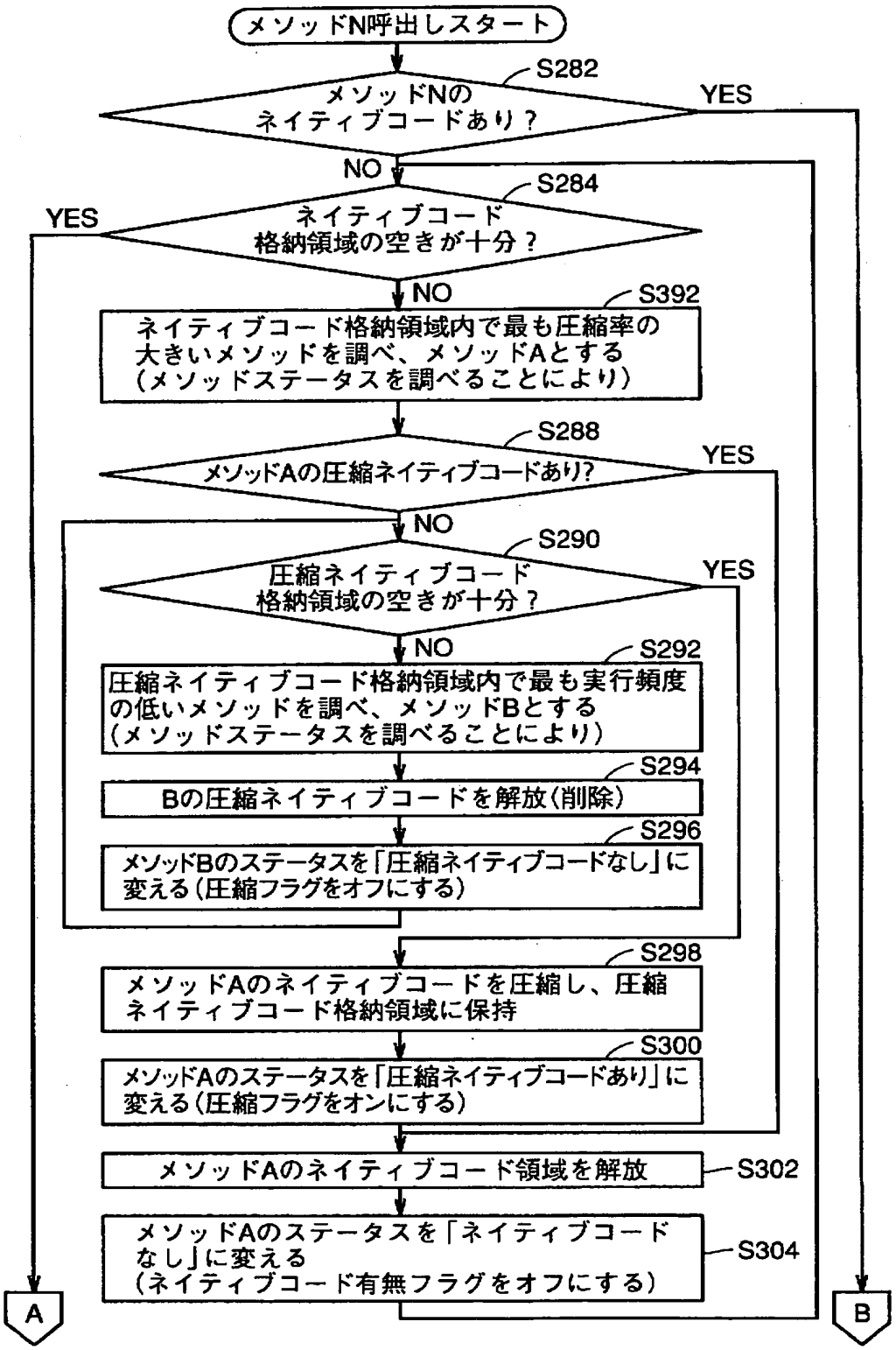
【図３０】


メソッドN呼出しスタート
S282
メソッドNの
ネイティブコードあり？
YES
NO
S284
ネイティブコード
格納領域の空きが十分？
YES
NO
S382
ネイティブコード格納領域内で最もサイズの
大きいメソッドを調べ、メソッドAとする
（メソッドステータスを調べることにより）
S288
メソッドAの圧縮ネイティブコードあり?
YES
NO
S290
圧縮ネイティブコード
格納領域の空きが十分？
YES
NO
S292
圧縮ネイティブコード格納領域内で最も実行頻度
の低いメソッドを調べ、メソッドBとする
（メソッドステータスを調べることにより）
S294
Bの圧縮ネイティブコードを解放（削除）
S296
メソッドBのステータスを「圧縮ネイティブコードなし」に
変える（圧縮フラグをオフにする）
S298
メソッドAのネイティブコードを圧縮し、圧縮
ネイティブコード格納領域に保持
S300
メソッドAのステータスを「圧縮コードあり」に
変える（圧縮フラグをオンにする）
メソッドAのネイティブコード領域を解放
S302
メソッドAのステータスを「ネイティブコード
なし」に変える
（ネイティブコード有無フラグをオフにする）
S304
A
B

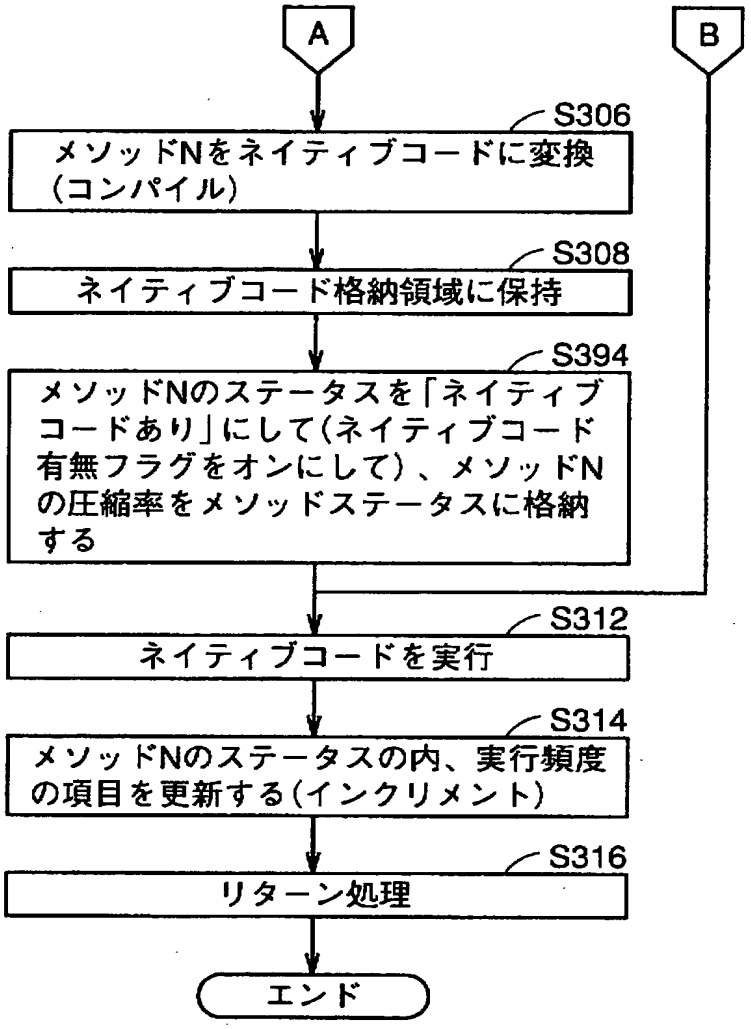
【図３１】


A
B
S306
メソッドNをネイティブコードに変換（コンパイル）
S308
ネイティブコード格納領域に保持
S384
メソッドNのステータスを「ネイティブコードあり」にして（ネイティブコード有無フラグをオンにして）、メソッドNのネイティブコードのサイズをメソッドステータスに記憶する
S312
ネイティブコードを実行
S314
メソッドNのステータスの内、実行頻度の項目を更新する（インクリメント）
S316
リターン処理
エンド

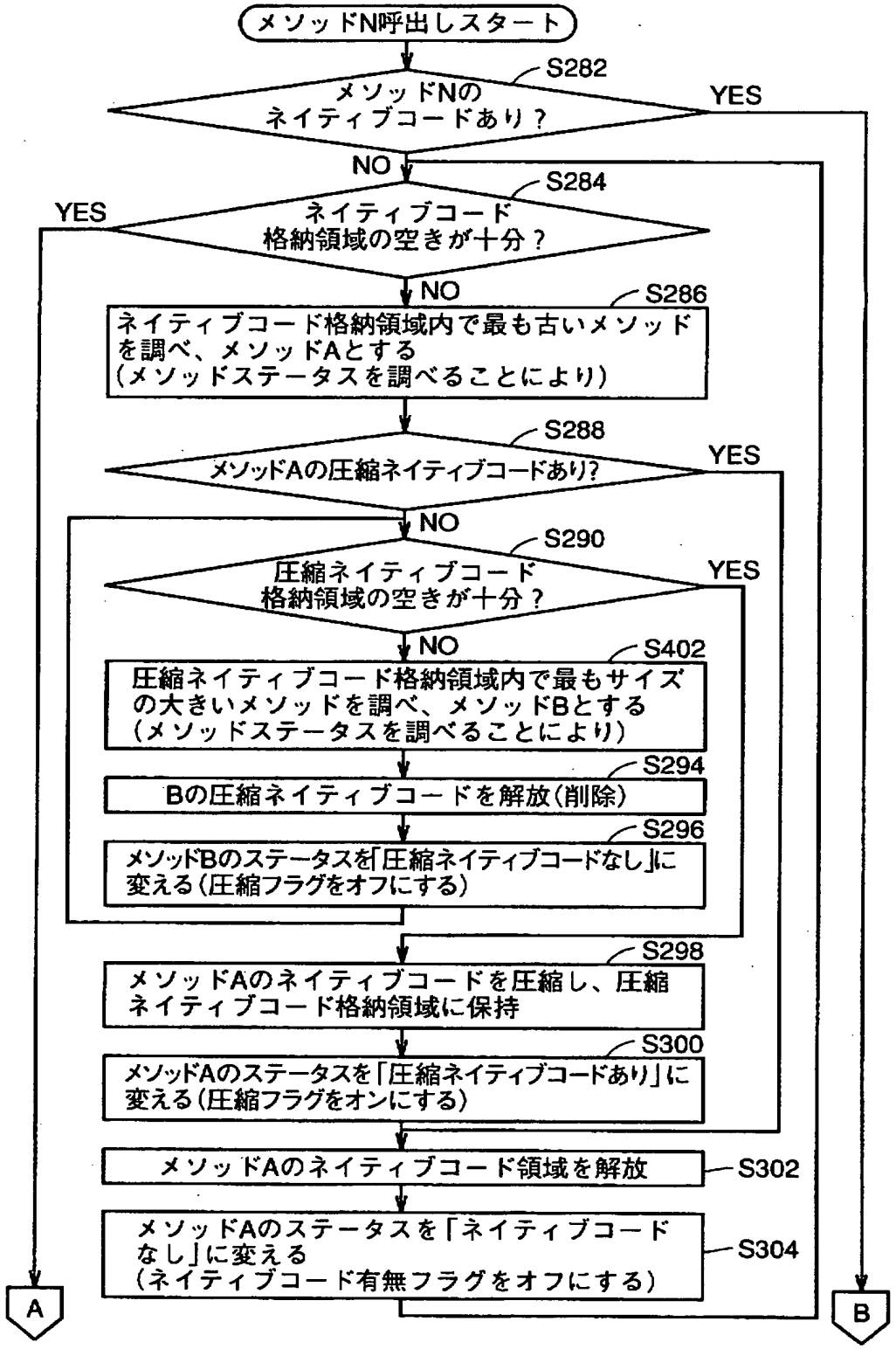
【図３２】


メソッドN呼出しスタート
S282
メソッドNの
ネイティブコードあり？
YES
NO
S284
ネイティブコード
格納領域の空きが十分？
YES
NO
S392
ネイティブコード格納領域内で最も圧縮率の
大きいメソッドを調べ、メソッドAとする
（メソッドステータスを調べることにより）
S288
メソッドAの圧縮ネイティブコードあり?
YES
NO
S290
圧縮ネイティブコード
格納領域の空きが十分？
YES
NO
S292
圧縮ネイティブコード格納領域内で最も実行頻度
の低いメソッドを調べ、メソッドBとする
（メソッドステータスを調べることにより）
S294
Bの圧縮ネイティブコードを解放（削除）
S296
メソッドBのステータスを「圧縮ネイティブコードなし」に
変える（圧縮フラグをオフにする）
S298
メソッドAのネイティブコードを圧縮し、圧縮
ネイティブコード格納領域に保持
S300
メソッドAのステータスを「圧縮ネイティブコードあり」に
変える（圧縮フラグをオンにする）
メソッドAのネイティブコード領域を解放
S302
メソッドAのステータスを「ネイティブコード
なし」に変える
（ネイティブコード有無フラグをオフにする）
S304
A
B

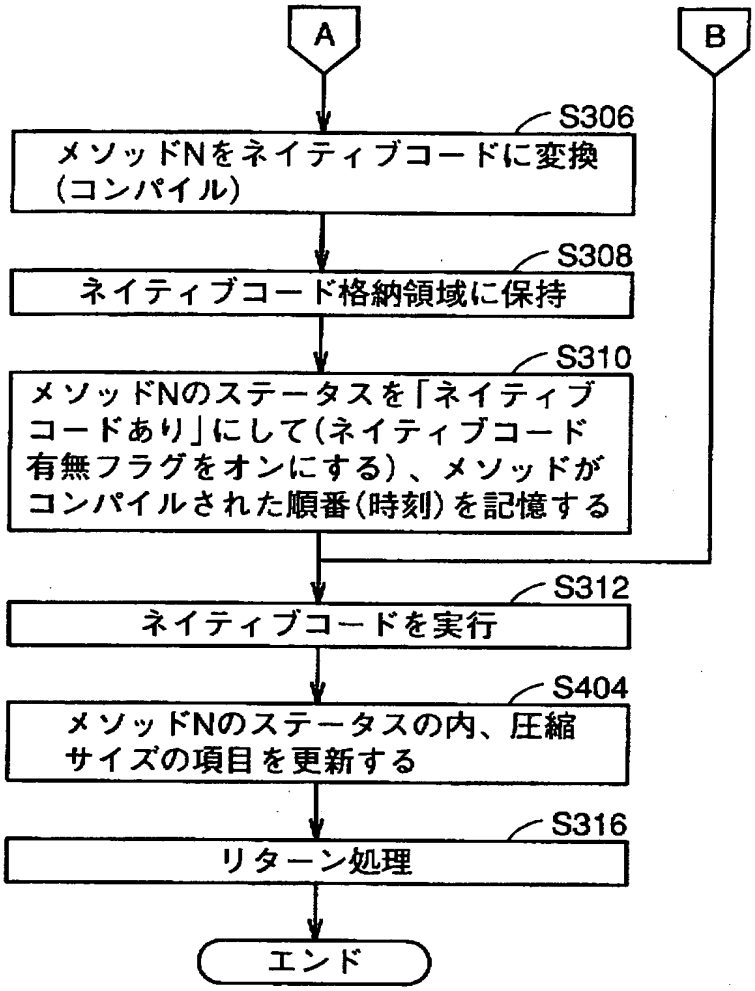
【図３３】


A
B
S306
メソッドNをネイティブコードに変換（コンパイル）
S308
ネイティブコード格納領域に保持
S394
メソッドNのステータスを「ネイティブコードあり」にして（ネイティブコード有無フラグをオンにして）、メソッドNの圧縮率をメソッドステータスに格納する
S312
ネイティブコードを実行
S314
メソッドNのステータスの内、実行頻度の項目を更新する（インクリメント）
S316
リターン処理
エンド

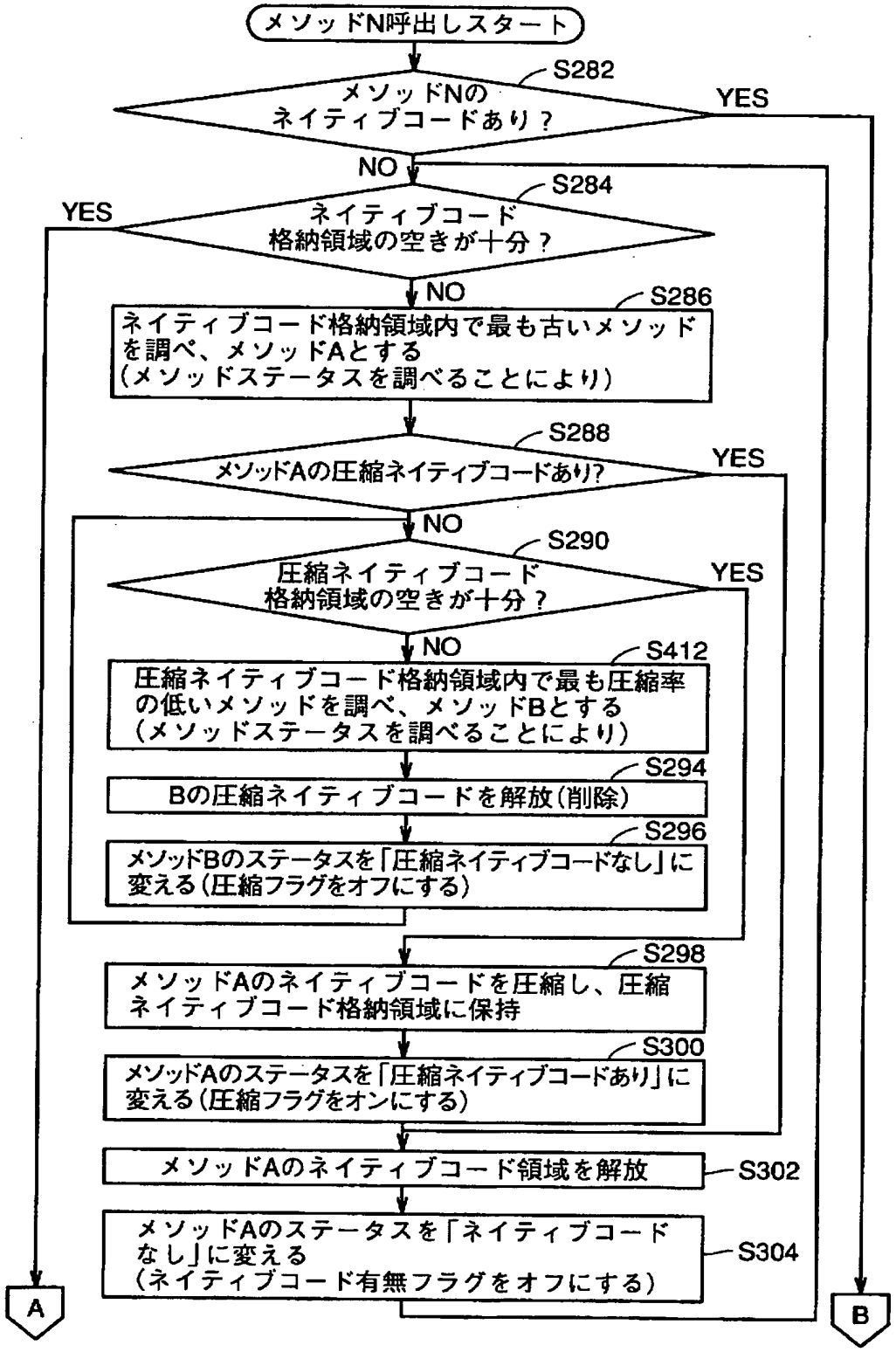
【図３４】


メソッドN呼出しスタート
S282
メソッドNの
ネイティブコードあり？
YES
NO
S284
YES
ネイティブコード
格納領域の空きが十分？
NO
S286
ネイティブコード格納領域内で最も古いメソッド
を調べ、メソッドAとする
(メソッドステータスを調べることにより)
S288
メソッドAの圧縮ネイティブコードあり?
YES
NO
S290
圧縮ネイティブコード
格納領域の空きが十分？
YES
NO
S402
圧縮ネイティブコード格納領域内で最もサイズ
の大きいメソッドを調べ、メソッドBとする
(メソッドステータスを調べることにより)
S294
Bの圧縮ネイティブコードを解放(削除)
S296
メソッドBのステータスを「圧縮ネイティブコードなし」に
変える(圧縮フラグをオフにする)
S298
メソッドAのネイティブコードを圧縮し、圧縮
ネイティブコード格納領域に保持
S300
メソッドAのステータスを「圧縮ネイティブコードあり」に
変える(圧縮フラグをオンにする)
メソッドAのネイティブコード領域を解放
S302
メソッドAのステータスを「ネイティブコード
なし」に変える
(ネイティブコード有無フラグをオフにする)
S304
A
B

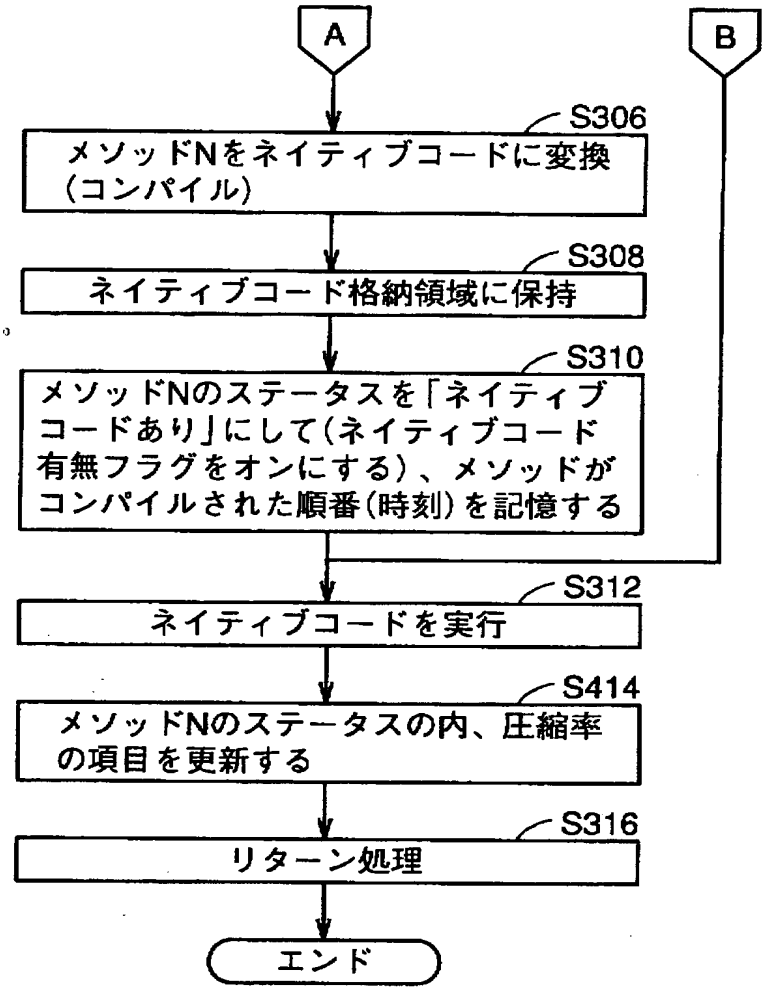
【図３５】


A
B
S306
メソッドNをネイティブコードに変換(コンパイル)
S308
ネイティブコード格納領域に保持
S310
メソッドNのステータスを「ネイティブコードあり」にして(ネイティブコード有無フラグをオンにする)、メソッドがコンパイルされた順番(時刻)を記憶する
S312
ネイティブコードを実行
S404
メソッドNのステータスの内、圧縮サイズの項目を更新する
S316
リターン処理
エンド

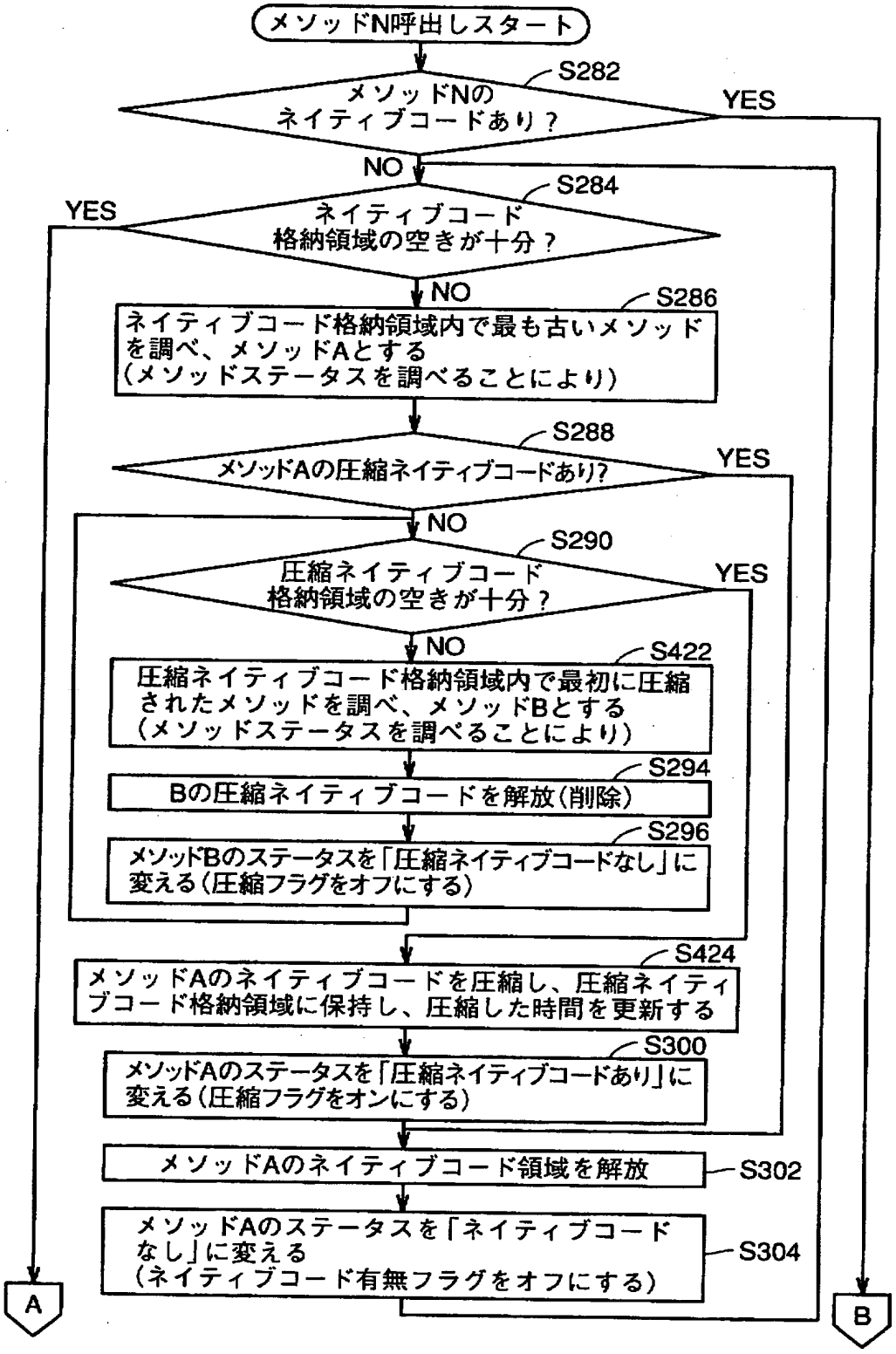
【図３６】


メソッドN呼出しスタート
S282
メソッドNの
ネイティブコードあり？
YES
NO
S284
ネイティブコード
格納領域の空きが十分？
YES
NO
S286
ネイティブコード格納領域内で最も古いメソッドを調べ、メソッドAとする
(メソッドステータスを調べることにより)
S288
メソッドAの圧縮ネイティブコードあり?
YES
NO
S290
圧縮ネイティブコード
格納領域の空きが十分？
YES
NO
S412
圧縮ネイティブコード格納領域内で最も圧縮率の低いメソッドを調べ、メソッドBとする
(メソッドステータスを調べることにより)
S294
Bの圧縮ネイティブコードを解放(削除)
S296
メソッドBのステータスを「圧縮ネイティブコードなし」に変える(圧縮フラグをオフにする)
S298
メソッドAのネイティブコードを圧縮し、圧縮ネイティブコード格納領域に保持
S300
メソッドAのステータスを「圧縮ネイティブコードあり」に変える(圧縮フラグをオンにする)
メソッドAのネイティブコード領域を解放
S302
メソッドAのステータスを「ネイティブコードなし」に変える
(ネイティブコード有無フラグをオフにする)
S304
A
B

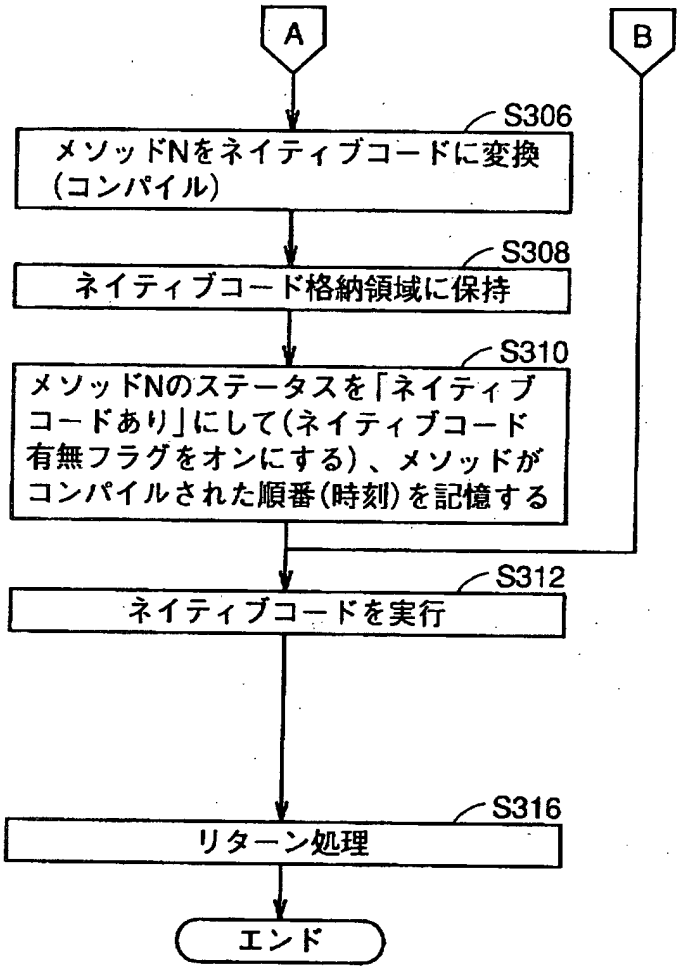
【図３７】


A
B
S306
メソッドNをネイティブコードに変換（コンパイル）
S308
ネイティブコード格納領域に保持
S310
メソッドNのステータスを「ネイティブコードあり」にして（ネイティブコード有無フラグをオンにする）、メソッドがコンパイルされた順番（時刻）を記憶する
S312
ネイティブコードを実行
S414
メソッドNのステータスの内、圧縮率の項目を更新する
S316
リターン処理
エンド

【図３８】


メソッドN呼出しスタート
S282
メソッドNの
ネイティブコードあり？
YES
NO
S284
ネイティブコード
格納領域の空きが十分？
YES
NO
S286
ネイティブコード格納領域内で最も古いメソッドを調べ、メソッドAとする
(メソッドステータスを調べることにより)
S288
メソッドAの圧縮ネイティブコードあり?
YES
NO
S290
圧縮ネイティブコード
格納領域の空きが十分？
YES
NO
S422
圧縮ネイティブコード格納領域内で最初に圧縮されたメソッドを調べ、メソッドBとする
(メソッドステータスを調べることにより)
S294
Bの圧縮ネイティブコードを解放(削除)
S296
メソッドBのステータスを「圧縮ネイティブコードなし」に変える(圧縮フラグをオフにする)
S424
メソッドAのネイティブコードを圧縮し、圧縮ネイティブコード格納領域に保持し、圧縮した時間を更新する
S300
メソッドAのステータスを「圧縮ネイティブコードあり」に変える(圧縮フラグをオンにする)
メソッドAのネイティブコード領域を解放
S302
メソッドAのステータスを「ネイティブコードなし」に変える
(ネイティブコード有無フラグをオフにする)
S304
A
B

【図３９】


A
B
S306
メソッドNをネイティブコードに変換（コンパイル）
S308
ネイティブコード格納領域に保持
S310
メソッドNのステータスを「ネイティブコードあり」にして（ネイティブコード有無フラグをオンにする）、メソッドがコンパイルされた順番（時刻）を記憶する
S312
ネイティブコードを実行
S316
リターン処理
エンド

【書類名】　　要約書

【要約】

【課題】　必要なメモリの容量が小さいプログラム実行装置を提供する。

【解決手段】　上位のモジュールよりあるメソッドの呼出しがあった場合には、呼出されたメソッドのバイトコードが、伸張されており、伸張済みバイトコード格納領域に格納されているか否かを調べる（Ｓ２）。バイトコードが伸張されていなければ（Ｓ２でＮＯ）、圧縮バイトコード格納領域に格納されたバイトコードが伸張され（Ｓ４）、伸張後のバイトコードが伸張済みバイトコード格納領域に格納される（Ｓ６）。このメソッドの圧縮バイトコードが伸張済みであることを表わすため、伸張済みフラグの値がオンに変更される（Ｓ８）。インタプリタを用いて、伸張済みのバイトコードが１命令ずつ解釈されながら実行される（Ｓ１０）。その後、呼出し側のモジュールへ復帰するための処理を行なう（Ｓ１２）。

【選択図】　　図５

出願人履歴情報

識別番号　〔０００００６０１３〕

１．変更年月日　１９９０年　８月２４日
［変更理由］　新規登録
住　所　東京都千代田区丸の内２丁目２番３号
氏　名　三菱電機株式会社